Spring 2-1-2002

SoProMaTT - A Software Project Management Teaching Tool

Sreedhar Thota
Dakota State University

SoProMaTT - A Software Project Management Teaching Tool

By: Sreedhar Thota

A Project submitted in partial fulfillment of the

requirements for the Master of Science in Information Systems

Dakota State University

2002

MSIS
PROJECT APPROVAL FORM

Student Name: Sreedhar Thota

Expected Graduation Date: December 2002

Master's Project Title: Sapromatt

Date Project Plan Approved: _____

Date Project Coordinator Notified and Grade Submitted: _____

Approvals/Signatures:

Student: _____ Date: 28 Feb. 03

Faculty supervisor: _____ Date: Feb. 28, 2003

Committee member: _____ Date: 2-26-03

Committee member: N/A Date: _____

Copies to:
Original Attached to Written Report
Copies to: Advisor, Graduate Coordinator, and Student

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Abstract

Software Project Management skills are vital to the success of a software project. The industry needs well-trained project managers to manage the complexities of software development processes. Aspiring project managers need to develop the management - related knowledge and skills. University education, in particular, needs to provide to their computer science and information systems students not only technology - related skills but also a basic understanding of typical phenomena occurring in software projects.

The objective of this thesis work is to develop a web - enabled teaching tool for software project management. Software Project Management Teaching Tool (SoProMaTT) is based on system dynamics model of software projects. It illustrates the consequences of software project management decisions. Students can run the tool using standard web browsers. The tool has two components - a user interface which can be a single user client machine or an intranet and a simulation component. The simulation component of the tool is implemented using system dynamics modeling with the help of the Extend Package. The student will be able to enter data, schedule and modify projects, and customize and produce reports. The tool provides analysis and decision making capabilities, which are necessary for effective project management. Possibilities for validation of the effectiveness of the tool in university education are described and future extensions to the tool are suggested.

CHAPTER 1 – Introduction

1.1 Background and Motivation

In practice, software project management is considered more as an art rather than a discipline. This view is supported by the fact that a software project's successful completion relies heavily on the project manager's expertise and experience to handle today's software projects, which are complex and dynamic in nature.

Also software projects are subjected to the multiple constraints of quality, scope and time to market. These factors make software project management a very difficult task in any organization. Some of the job functions of a project manager include:

- Define the scope of the project
- Identify stakeholders, decision-makers, and escalation procedures
- Develop detailed task list
- Estimate time requirements
- Identify required resources and budget
- Evaluate project requirements
- Identify and evaluate risks

The typical project manager in an organization fits into the following categories:

- A person who has sufficient technical expertise and has risen to the rank of a project manager from being a team member of a project. These people are well trained in the software development life cycle and the constituent software engineering processes, but may not have enough managerial and people skills required to manage a software project.

- A person who is a functional head with more management skills and less of technical expertise such as software development skills and quality assurance skills.

Both these types of project managers do have their strengths and weaknesses. Of course experience and the individual skills of these people do matter in determining the project's success. An ideal project manager would have enough expertise in both areas. A good project manager needs management skills, strong communication, leadership, and political skills. They also need skills in organization, teamwork, coping, and making effectively use of technology.

The industry needs such experienced and well-trained project managers. Software engineering, computer science and Information Systems students are taught the concepts of project management. The focus is on the theoretical aspects of the discipline. Very less practical exposure is given to the students. This is due to the fact that to get hands on experience and to get a feel of the industry requires the students to actually manage of a software project. This is not feasible in most cases. This is where simulation steps in. Effective simulation of the development scenario can help students and other aspiring project managers overcome this handicap.

Much work has been done in the system dynamics field to create effective simulation environments for project management. (Tarek-Abdel Hamid (1991), Sycamore (1996), Collafello (1997), Dietmar (2000) and others to name a few). Also experiments have been conducted at the Jet Propulsion Laboratory and Draper Laboratory to study the decision making process of software managers and to study the effect of applying requirement changes in the course of a simulated project. Although

these experiments focused on exposing people to realistic problems through the use of simulated models, they did not fully focus on analysis of errors and feedback.

Another important work was undertaken at the Fraunhofer IESE by Dietmer Pfahl, Marco Klemm, and Gunther Ruhe (2000). A computer based model for student education in software project management was built by the Institute. The focus was to transfer knowledge about software project management to computer science and Software Engineering students by providing a scenario-driven interactive single learner environment that can be activated through a standard web browser. The tool developed by the institute was not fully web enabled. The trainee would have to switch between the web browser and the simulation tool.

The main objective of this work is to first incorporate system dynamics models to simulate the software development scenario in the teaching tool and secondly provide a web enabled interface for the tool so that students can access the tool over the intranet and it can be a multi user environment.

1.2 Objective

The aim of this work is to train university students the managerial aspects of software project management by using simulation models. These simulation models expose aspiring managers to real life scenarios, and allow them to develop experience without the risks involved in the real world. The second objective of this work is to build a web interface for the tool. The tool has to be deployed on the campus intranet and students can access the tool from anywhere within the university campus. It also has to be modular and scalable. It should expose the students to realistic project management scenarios and be able to give effective feedback to the students.

CHAPTER 2 - Literature Review

2.1 Introduction

This chapter summarizes the previous work done in the field of system dynamics and simulation for project management and project management education is presented. Section 2.2 discusses project management issues and characteristics. Section 2.3 gives an overview of system dynamics. Section 2.4 talks about work done in the field of system dynamics and simulation for software project management. Finally, section 2.5 discusses the work done in the field of system dynamics and simulation for project management education.

2.2 Project management principles, issues and characteristics

A successful software project requires

- Understanding the scope of the work that needs to be done
- Identifying the risks in advance
- Acquiring the resources
- Accomplishing the activities
- Tracking the milestones
- Calculating the effort to be spent
- Determining the schedule to be followed (Pressman, 1992).

Project management knowledge and practices are best described in terms of their component processes. These processes can be placed into five Process Groups (PMBOK, 2000):

- Initiating
- Planning

- Executing
- Controlling
- Closing

Nine Knowledge Areas:

- Project Integration Management,
- Project Scope Management,
- Project Time Management,
- Project Cost Management,
- Project Quality Management,
- Project Human Resource Management,
- Project Communications Management,
- Project Risk Management,
- Project Procurement Management.

The techniques currently applied to software project management are dated from the 1950's and 1960's, developed to allow the accomplishment of advanced military projects, such as the Polaris project. These techniques include work breakdown structures, PERT/CPM charts, Gantt diagrams, and crashing. They are based on a set of fundamental assumptions, necessary for their accuracy (Charette, 1996). Some of these techniques require that projects have clear and delimited objectives, there exists at least one solution to the problem at hand, development time and resources can be precisely stated before the project starts, the operational environment is well known and defined, and that quality metrics can be quantified for the project. Due to many successes resulted from using these techniques in large projects, project managers tend to passively take

their underpinning assumptions for granted in every project (Charette, 1996). This misconception is very common in software project management. Current project management techniques' baseline assumptions require a project behavior to be known from the beginning.

Modern projects are characterized by constantly changing requirements, ambiguity, complexity, discontinuity, and diseconomy of scale, nonlinearities, and complex feedback loops. Traditional techniques are not entirely sufficient to tackle all these issues. This is where simulation and system dynamics step in. System Dynamics can be an effective tool to study the implications of the various managerial decisions involved in a typical software project.

2.3 System dynamics overview

System Dynamics: "System Dynamics deals with the time-dependent behavior of managed systems with the aim of describing the system and understanding, through qualitative and quantitative models, how information feedback governs its behavior, and designing robust information feedback structures and control policies through simulation and optimization."

System dynamics was developed 40 years ago by Jay Forrester at MIT to improve organizational structures and processes, but is just recently being applied in software engineering settings. It provides an integrative framework for capturing different process phenomena and their relationships. Models are formulated using continuous quantities interconnected in loops of information feedback and circular causality. The quantities are expressed as levels (stocks or accumulations), rates (flows) and information links representing the feedback loops (Madachy and Boehm, 2001)

System dynamics is characterized by:

- Searching for useful solutions to real problems, especially in social systems (businesses, schools, governments) and the environment.
- Using computer simulation models to understand and improve such systems.
- Basing the simulation models on mental models, qualitative knowledge and numerical information.
- Using methods and insights from feedback control engineering and other scientific disciplines to assess and improve the quality of models.
- Seeking improved ways to translate scientific results into actual implementation.

2.4 System dynamics and project management

The application of System Dynamics (SD) to Project Management has become increasingly important during the last decade. The application of SD to Project Management covers a wide range of uses, in particular creating team learning and training environments, providing a tool for advanced planning and control of on-going projects. Much work has been done in the system dynamics field to create effective simulation environments for project management.

System dynamics modeling was applied to the software development process for the first time by Abdel Hamid and Madnick (1992). They developed a model that captures the managerial aspects of a waterfall software life cycle. They studied the impact of turnover, acquisition, and assimilation rates on software project cost and schedule.

Sycamore (1996) developed a software management tool developed using equations that represent causal influences rather than statistical correlation. He calculated the results using a system dynamic model developed. His model takes into consideration

areas like communication overhead, schedule pressure, and rework hours generated by defects. Before a software manager makes a modification to a project, this tool can be used to simulate the modification by adjusting the staffing profile, the expertise of the staff, the dependencies of the software increments, and many other features and visually observing the predicted outcome. As the simulation runs, many different outputs are updated in a real-time fashion to the screen and the final results can then be logged for later analysis.

Tvedt developed a model that could enable researchers and software project managers to evaluate the impact, a set of process improvements will have on their software development cycle time. The model enables researchers and software project managers to gain insight and perform controlled experiments to answer "What if?" type questions, such as, "What kind of cycle time reduction can I expect to see if I implement inspections?" or "How much time should I spend on inspections?."

Experiments have been conducted at the Jet Propulsion Laboratory and Draper Laboratory to study the decision making process of software managers and to study the effect of applying requirement changes in the course of a simulated project. Although these experiments focused on exposing people to realistic problems through the use of simulated models, they did not fully focus on analysis of errors and feedback.

2.5 System dynamics and software project management education tools

Merrill and Collofello (1997) built a Software Project Simulator to training Software Development Project Managers. Their goal was to create an effective simulation tool and training process for software project management training. The tool was built upon a model of the software development process and exposed the trainee to

realistic software project management training situations for which a manager must plan and react. A training process accompanied the tool to provide a more complete method through which specific software project management lessons could be taught and explored.

Another important work was undertaken at the Fraunhofer IESE by Dietmer Pfahl, Marco Klemm, and Gunther Ruhe (2000). A computer based model for student education in software project management was built by the Institute. The focus was to transfer knowledge about software project management to computer science and Software Engineering students by providing a scenario-driven interactive single learner environment that can be activated through a standard web browser.

This computer based training (CBT) module for student education in software project management has a web interface for a component that imparts theoretical aspects of project management. The simulation component of the CBT module is implemented using the System Dynamics simulation modeling method. This component is a stand-alone component and it provides a single learner environment. The tool developed by the institute was not fully web enabled. The student has to switch between the web enabled component and the stand-alone component.

CHAPTER 3 - System Design

3.1 Introduction

The purpose of this chapter is to present an overall description of the software project management teaching tool (SoProMaTT). The basic design issues are introduced in Section 3.2. Section 3.3 is a general description of SoProMaTT while section 3.4 lists SoProMaTT's major features and specifications. Section 3.5 introduces the issues considered in the choice of the development tools. The last section in this chapter, section 3.6 lists the architecture of the tool.

3.2 Basic Design Issues

Creating an effective simulative environment of the software development scenario poses a lot of challenges. Thus to teach software project management the concepts have to be broken down into manageable lessons or modules. Hence a modular design with well defined interfaces between the various modules is an important design issue that should be taken into consideration.

Cost is also an important design issue as it has a direct implication on the economic feasibility of the teaching tool. For any university to design, develop, and deploy this tool cost has to be taken into consideration. The factors that could increase the cost of the teaching tool are the licensing costs associated with deploying it over the intranet or for that matter the internet.

The next design issue is flexibility. Flexibility means the ease of modifying the tool to accommodate different scenarios and newer issues. Flexibility and modularity go hand in hand as a modular design can easily accommodate for future changes or additions to the tool.

Scalability and generality are other design issues, which concern the applicability of the teaching tool to various subjects. Although the tool is primarily intended for students, it can cater to the needs of aspiring project managers from the industry. This can be achieved by a modular and flexible design.

A web interface is essential as the tool can then be accessed from anywhere within the campus intranet. The tool should also have an easy friendly user interface to enable the tool to be efficiently used.

Last but not the least; the teaching tool has to be reliable. The models that form the heart of the tool have to simulate the software development scenario realistically. Reliability implies the ability of the models to offer results that the student can rely on.

3.3 System Description

The objectives of the proposed tool (SoProMaTT) are

- To enable students to get a feel of the software development scenario and issues through simulation.
- To have a web enabled user interface

However due to the wide range of development issues involved, a problem solving methodology is needed. The problem solving methodology employed is broken down in two steps:

- Step 1: Here the primary concern is to develop the simulation models. The simulation models need to be accurate and reliable and should be able to capture the level of detail expected.

- Step 2: The focus here is to provide a web interface to the models developed in Extend. At this stage the selection of developmental tools and the development methodology needs to be evaluated.

SoProMaTT includes two main components: a simulation modeling component, and a programming and user interface component as shown in figure 3.1. The arrows shown in the figure denote possible interactions, in which the student, through the user interface component, can access the modeling component. The modeling component is composed primarily of simulation models. figure 3.2 shows the present models incorporated in SoProMaTT. The models are:

1. Brooks Law model
2. Rayleigh's staffing curve model
3. Quality and the effect of Inspections model

The user interface component of the tool is a web interface. The student should be able to enter the data through this component and these inputs are then sent to the simulation models. The dialog component of he blocks available in the simulation models facilitates the input.

3.4 System features and specifications

The special features and characteristics of SoProMaTT include:

1. Modularity:

This is attained by designing the system in modules with clear and well defined interfaces. This modular design approach has contributed significantly to lowering the system's development time and thus cost, as well as facilitating future expandability of the system.

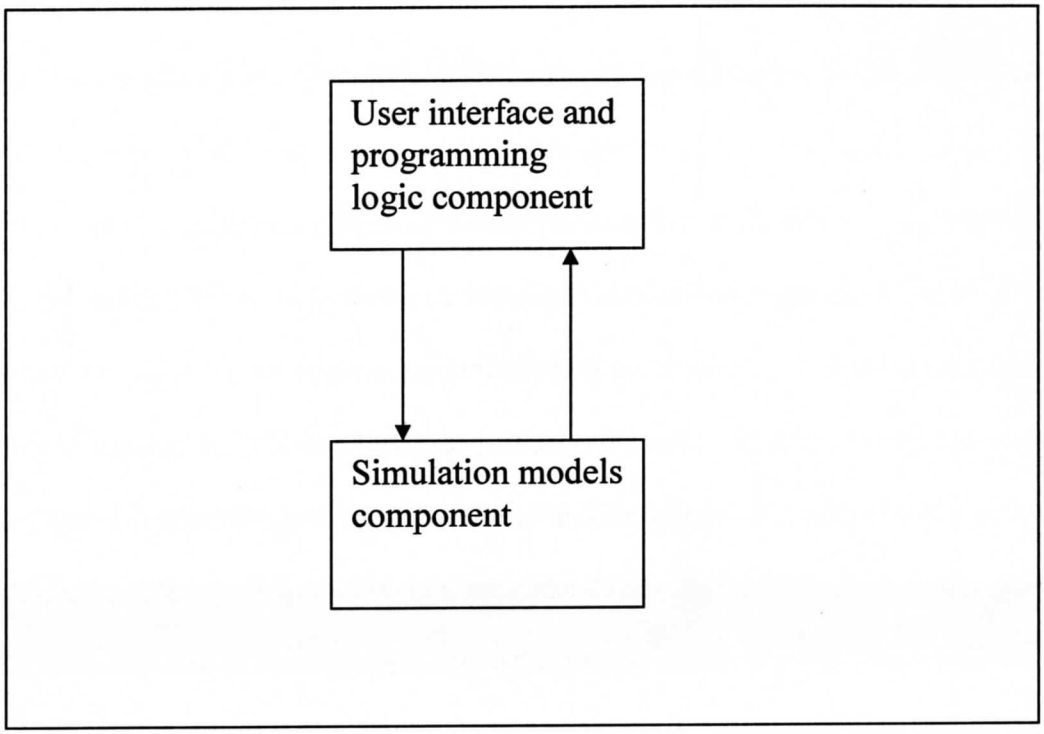

Figure 3.1. SoProMaTT Framework

2. **Low cost:**

The tool is built by using development tools that are not expensive. The simulation component is built by using the Extend package which provides features for continuous and discrete modeling as well as providing tools for interfacing with Visual Basic. The deployment is on the server and since a web interface is used no additional costs are incurred.

3. **Flexibility:**

This is attained by designing the two components of SoProMaTT, namely the simulation models component, and the user interface component to be loosely coupled with each other. Thereby facilitating the modification of the tool to accommodate for future changes.

4. **Scalability and generality:**

This is mainly achieved through the modular and flexible design which would enable expanding SoProMaTT to include more models in the future to deal with different software project management scenarios.

5. **Web based user interface:**

The tool is built with a web interface. It is deployed on the campus intranet and this makes it accessible to all the students in the campus. Also being web enabled a URL and a web browser is needed to access the tool.

6. **Reliability**

The results obtained are reliable and accurate and they tally with the results obtained by using models developed using other simulation packages demonstrating the various phenomena associated with software development.

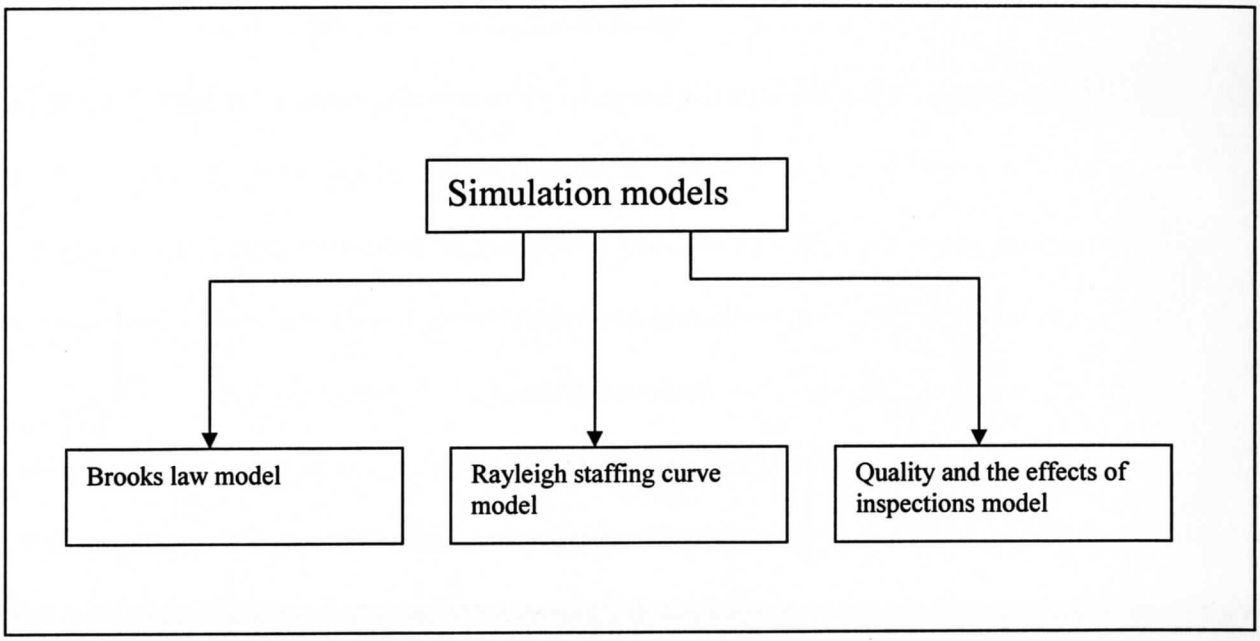

Figure 3.2. The modeling component

3.5 Development tools

SoProMaTT has two components: the simulation models component and the user interface component. Both these component have their own processing requirements as well as their own development environments.

With regard to the simulation models component, the factors that had to be considered in choosing the package were:

- The package should support system dynamics modeling
- The package had to be interfaced with some client side development tool in order to build a front end.

With these considerations in mind Extend V5 was chosen. Extend supports both continuous and discrete event modeling. The continuous modeling aspect of Extend could be used for developing the system dynamics models. Also Extend has a set of DLLs (dynamic linked libraries) that allow it to be interfaced with Visual Basic and C++. In other words Extend can be used as a server and the GUI (Graphical user interface) could be built using Visual basic or C++.

For the programming logic and user interface component Visual Studio 6.0 was chosen as Visual Basic could be used as the client side development language. In our case since we need to web enable the tool, the IIS web application template of Visual Basic 6.0 is selected for building the programming logic and the user interface.

Table 3.1 summarizes the architecture of the tool:

Component	Implementation
Hardware: Client	400 MHz Pentium based client with 256MB RAM and 60 GB Hard Disk
Hardware: Server	400 MHz Pentium based client with 256 MB RAM and 60 GB Hard Disk
Software: Operating System(server)	Windows 2000
Software: Operating System(client)	Windows 2000
Software: Application(client)	VB 6.0 Enterprise edition
Software: Middle Tier	Extend model and DLL's
Software: WEB (server)	IIS
Software: WEB interface (server)	IIS application
Software: Visual Modeling	MS- Visual Modeler
Protocol: Network	TCP/IP

CHAPTER 4 - The Anatomy of the system

4.1 Introduction

This section describes SoProMaTT in detail. Section 4.2 deals with the Simulation models component, where each model is explained from the user and the developer point of view. Section 4.3 discusses the programming logic and the user interface component of the tool. Section 4.4 describes how the tool is packaged in the form of lessons and how the lessons incorporate the simulation models.

4.2. The simulation models component

The simulation component consists of three models namely the Brooks Law model, the Rayleigh staffing curve model, and the Quality Assurance model. These models are developed using the Extend package. The models are described in detail below.

4.2.1 Brooks Law model

In the book The Mythical Man-Month, Fred Brooks (1975) stated *"Adding manpower to a late software project makes it later"*. His explanation for the law was the additional linear overhead needed for training new people and the nonlinear communication overhead (a function of the square of the number of people). These effects have been widely accepted and observed by others. The simple model in Figure 4.1 describes the situation, and will be used to test the law. This model demonstrates the effect of adding manpower to a project at various phases.

Brooks law explains the reason why adding people to a late project makes it later.

Whenever a software project shows signs of not meeting the schedule project managers tend to add new personnel to speed up the project. But this has an adverse effect on the project and actually delays further. Brooks attributes this to the following reasons:

- Experienced developers must reduce their work rate to train new personnel
- As a team grows in size, effort wasted in communication increases
- communication losses are expected to be proportional to n(n-1) where n is the number of people

The system dynamics model can be broken down to two subsystems:

1. work progress
2. human resource

Suppose we know (or have estimated) the size of the project requirements or initial amount of work to be completed. In this case, we have expressed the work to be completed in Function Points units (one could also use lines of code). The initial amount of work to be completed (requirements) is 500 FP. We are interested in the amount of work to be completed as a function of time.

The rate_of_change of work_to_be_completed (also known as productivity) depends on:

- The nominal_productivity: the number of Function Points a single team member can implement in a day of work (PF/person-day). For this problem, nominal_productivity is 0.3 FP/day.
- The number of people N on the team.

As a first approximation, the rate-of-change is proportional to the number of people N on the development team. The proportionality factor is the nominal_productivity. If one assumes every team-member communicates with every other team member (a clique),

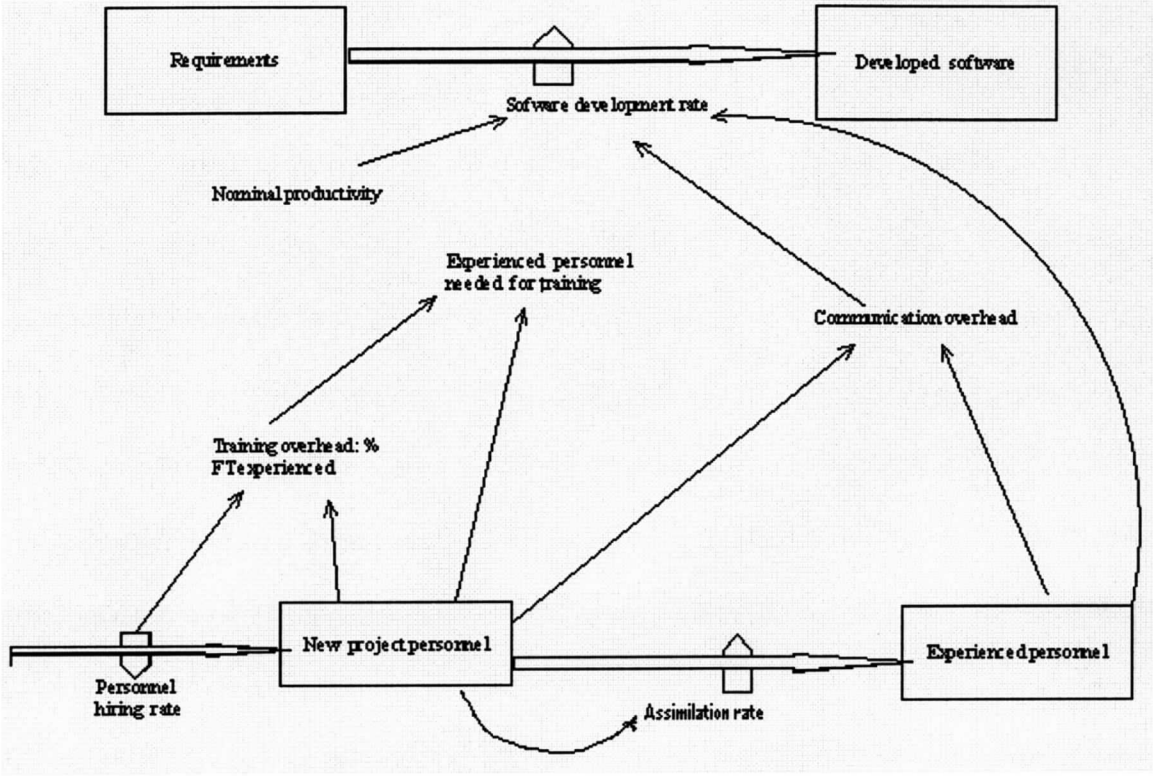

Figure 4.1 Brooks Law causal loop diagram

the amount of communication is proportional to N^2. The communication overhead proportionality factor is C_overhead (0.6% in our model). Communication is detrimental to productivity and will hence have a negative influence on the rate-of-change of work_to_be_completed

The equations for the model are:

Assimilation Rate = New Project Personnel/20 *(4.1)*

It is the rate in which the new personnel get assimilated into the project

This means that it takes 20 days for a new project personnel to become experienced personnel.

Communication Overhead = 0.06(Experienced Personnel + New Project Personnel)* ^{2} *(4.2)*

The amount of communication is proportional to the square of the number of team members present in the project. The communication overhead proportionality factor is 0.06.

Software Development Rate = Nominal Productivity(1-Communication Overhead/100)*(0.8*New Project Personnel+1.2*(Experienced Personnel-Experienced Personnel Needed for Training))* *(4.3)*

The rate at which software gets developed takes into account the nominal productivity, which is 0.3 FP in this case. The factor for new project personnel and experienced personnel is 80% and 120% of the Nominal Productivity.

4.2.2 Rayleigh Staffing curve model

Rayleigh curve is a popular model of personnel loading. The underlying assumptions are for the model are:

- Only a small number of people are needed at the beginning of a project to carry out planning and specification. As the project progresses and more detailed work is required, the number of staff builds up to a peak. After implementation and unit testing is complete, the number of staff required starts to fall until the product is delivered.
- The number of people working on a project is approximately proportional to the number of problems ready for solution at that time

Rayleigh Formula

A Rayleigh curve describes the rate of change of manpower effort per the following first order differential equation:

$$dC(t) / dt = p(t)[K - C(t)] \qquad (4.4)$$

where

- $C(t)$ is the cumulative effort at time t
- K is the total effort
- $p(t)$ is a learning function.

The learning function is linear and can be represented by

$$p(t) = 2at \qquad (4.5)$$

where

- a is a positive number.
- The manpower rate of change represents the number of people involved in development at any time (staffing profile).
- The a parameter is an important determinant of the peak personnel loading called the manpower buildup parameter.

Effort_rate = learning_function(estimated_total_effort – cumulative_effort)* *(4.6)*

The effort rate is calculated by considering the estimated total effort and the cumulative effort which is the effort completed at that point of time.

*Learning_function = manpower_buildup_parameter * time* *(4.7)*

Learning function is the rate at which the new personnel get added at that certain time.

Manpower_buildup_parameter = 0.5 *(4.8)*

The manpower buildup parameter determines the steepness of buildup and the peak personnel loading.

The graphs showing the Rayleigh curves for different values of a is shown in figure 4.3

4.2.3 Quality model

This model demonstrates the effect on increasing the number of inspections on the quality of the software project.

The student can use this model to vary the following:

- Decrease the functionality of the system
- Increase the number of inspections

By increasing the number of inspections the number of errors go up, thereby increasing the rework. As the rework increases the rate of developing the software also decreases and the time to develop the software also increases. However the quality of the system developed is higher.

When faced with a situation of meeting the schedule and not compromising the quality of the system the student can reduce the functionality of the product thereby ensuring good quality.

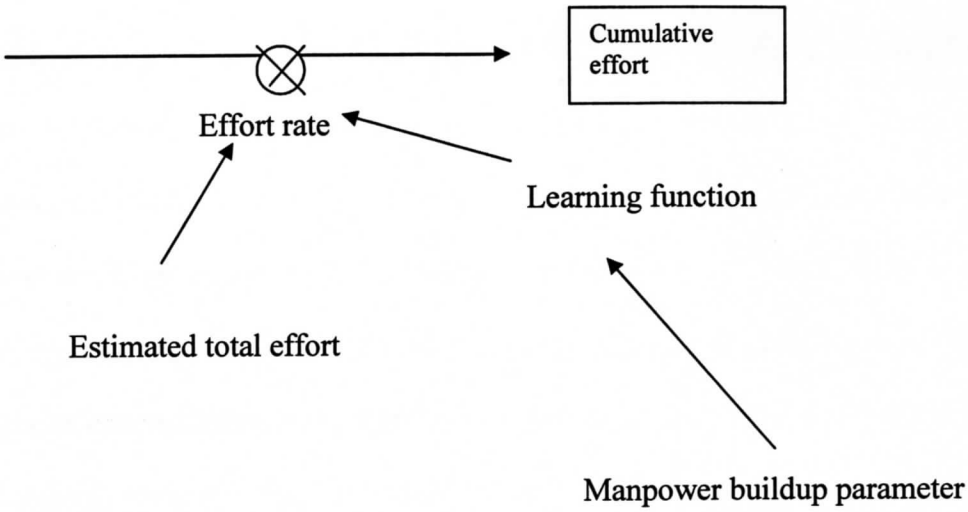

Figure 4.2 Rayleigh staffing curve causal diagram

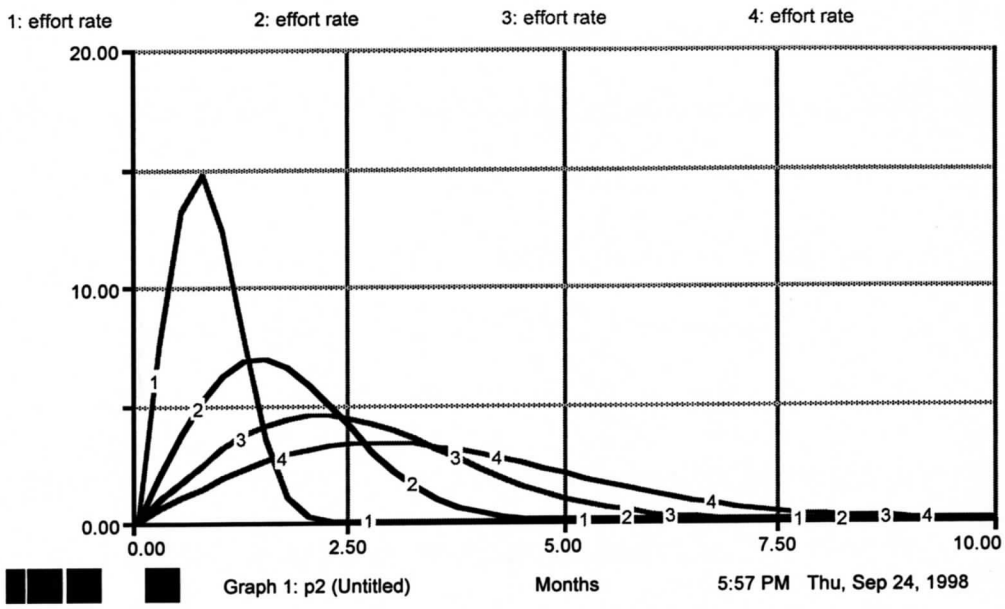

Figure 4.3 Graphs showing Rayleigh curves

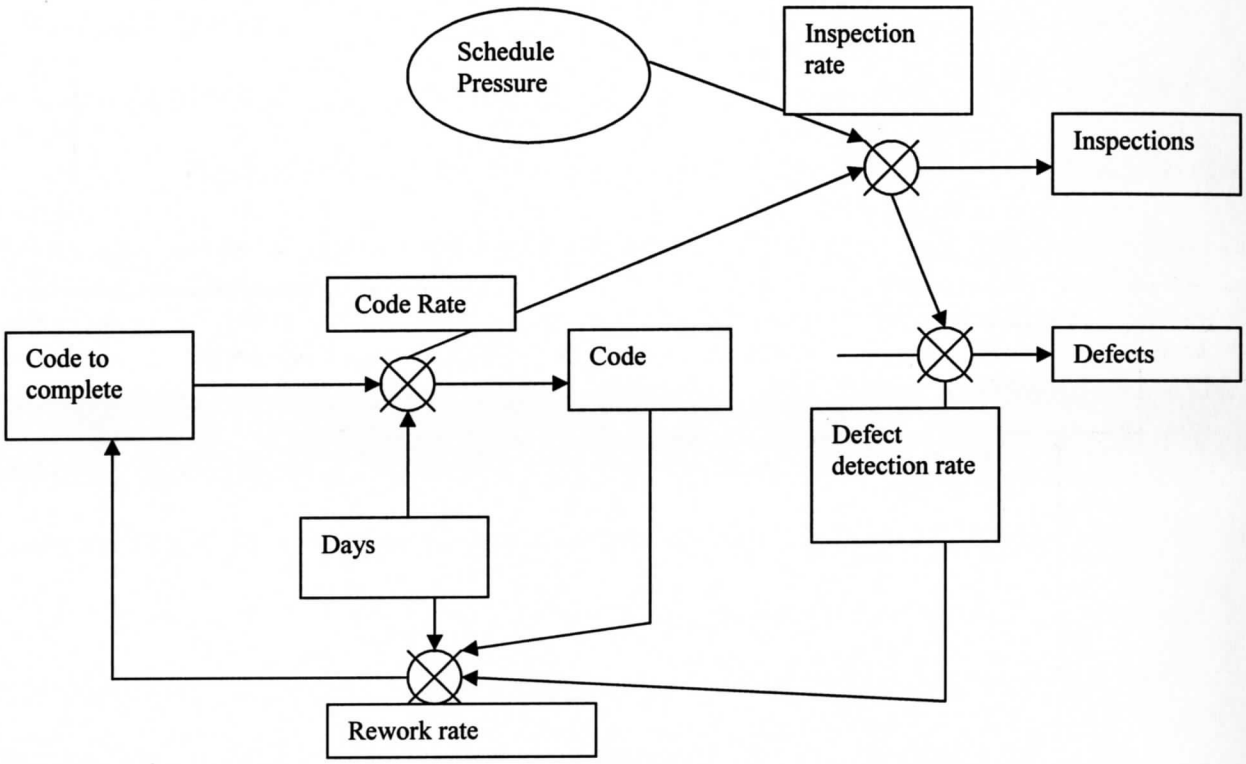

Figure 4.3 Quality causal loop diagram

4.3 Programming logic and user interface component

The simulation models developed using Extend produce results primarily in the form of graphs. The inputs to the models are done by using the dialog component of each block. Extend has dynamic link libraries that allow the models to be interfaced to programming languages such as Visual Basic and C++.

For this system Visual Basic 6.0 has been used to develop the screens and the programming logic. Extend is used as a server and Visual Basic is used as the client.

To web enable the tool The IIS web application template of Visual Basic 6.0 is used. The appendix has details regarding the implementation of the tool.

4.4 Lessons

SoProMaTT is organized in the form of lessons. Currently there are four lessons incorporated in the tool. Out of these, three lessons use the simulation models and one lesson deals with project management principles.

The lessons are as follows:

- Project Management principles and an overview of system dynamics
- Brooks Law
- Rayleigh's Staffing Curve
- Quality Assurance

4.4.1 Lesson 1: Project Management Principles

This module delivers the course content to the student in the form of static text, which is presented in a web-based form. It has an introduction to the duties of a software

project manager. It also describes the project management process group, the activities and knowledge areas of a software project.

The terminology used is explained and appropriate references in the form of books, manuals and links to useful web sites will be provided.

The student can start using the tool by first reading this lesson or use this for reference and lookup purposes when he accesses the subsequent lessons.

4.4.2 Lesson 2: Brooks Law

The second lesson contains the system dynamics model of Brooks Law, which is implemented using the Extend Package. The lesson is organized in the form of three sub topics:

- A study guide: The study guide has the objectives outlined. It shows the steps to be taken by the student to successfully use this module. It also has the links to the important references and a list of books that may be used by the student to get more understanding of Brooks Law.
- Lesson Notes: The Lesson notes explain Brook's Law and the system dynamics model in detail. Appendix 2 covers the model details and the explanation.
- Activities: The activities section is where the student runs the simulation model. He can change various parameters and see the effect by running the model.

  The input parameters are:

  - Initial Requirements
  - Experienced Personnel (Initial number of experienced personnel)
  - Rookies (Initial number of fresh personnel)
  - Time delay (After how many days are additional rookies added)

The output is a graph of software development rate versus time.

- Assignments: After performing the activities the student can begin the assignments. Here they are put to a test to what they have learnt using this module. The assignments are related to the study material and the results obtained by running the simulation model. The students can submit the assignments and then get the feedback from the faculty who grade the assignments.

4.4.3 Lesson 3: Rayleigh's Staffing curve

The third lesson contains the system dynamics model of Rayleigh's Staffing Curve. The lesson is organized in the form of three topics

- A study guide: The study guide has the objectives outlined. It shows the steps to be taken by the student to successfully use this module. It also has the links to the important references and a list of books that may be used by the student to get more understanding of Rayleigh's Staffing Curve.
- Lesson Notes: The Lesson notes explain Rayleigh's Staffing Curve and the system dynamics model in detail. Appendix 2 covers the model details and the explanation.
- Activities: The activities section is where the student runs the simulation model. He can change various parameters and see the effect by running the model.

  The input parameters are:

  - Estimated total effort
  - Manpower buildup rate

  The output is a graph of effort rate versus time.

- Assignments: After performing the activities the student can begin the assignments. Here they are put to a test to what they have learnt using this module. The assignments are related to the study material and the results obtained by running the simulation model. The students can submit the assignments and then get the feedback from the faculty who grade the assignments.

4.4.4 Lesson 3: Quality and the effect of inspections

The last lesson contains the system dynamics model of Quality and the effect of inspections. The lesson is organized in the form of three topics

- A study guide: The study guide has the objectives outlined. It shows the steps to be taken by the student to successfully use this module. It also has the links to the important references and a list of books that may be used by the student to get more understanding of Quality and the effect of inspections.
- Lesson Notes: The Lesson notes explain Quality and the effect of inspections and the system dynamics model in detail. Appendix 2 covers the model details and the explanation.
- Activities: The activities section is where the student runs the simulation model. He can change various parameters and see the effect by running the model.

  The input parameters are:

  - Experienced Personnel
  - Rookies
  - Inspection Rate

The outputs are:

- A graph of code developed versus time.
- A graph of inspections versus time.
- A graph of rework versus time.

Assignments: After performing the activities the student can begin the assignments. Here they are put to a test to what they have learnt using this module. The assignments are related to the study material and the results obtained by running the simulation model. The students can submit the assignments and then get the feedback from the faculty who grade the assignments.

CHAPTER 5 - System Testing and evaluation

5.1 Introduction

The testing and evaluation of the tool comprises of two steps. The first step is the testing of the tool. The second step is the evaluation mechanism. The first step has been concluded whereas the second step is in the planning phase.

5.2 Step 1 Testing the usability of SoProMaTT

Step 1 aims at testing the usability of SoProMaTT.

SoProMaTT has to be tested for reliability and accuracy. The models are put to test by comparing the results obtained by the simulation models developed in Extend to those developed in other simulation packages.

The graphs obtained by the models compare favorably to the one obtained by using other tools. Figure 5.1 shows a sample comparison obtained by running the Rayleigh staffing curve model.

Secondly as SoProMaTT has been developed as a web enabled tool it has to be put to the test whether it can be accessed from anywhere in the university campus.

This has been tested by launching SoProMaTT from the URL

C://localhost/ SoProMaTT.asp

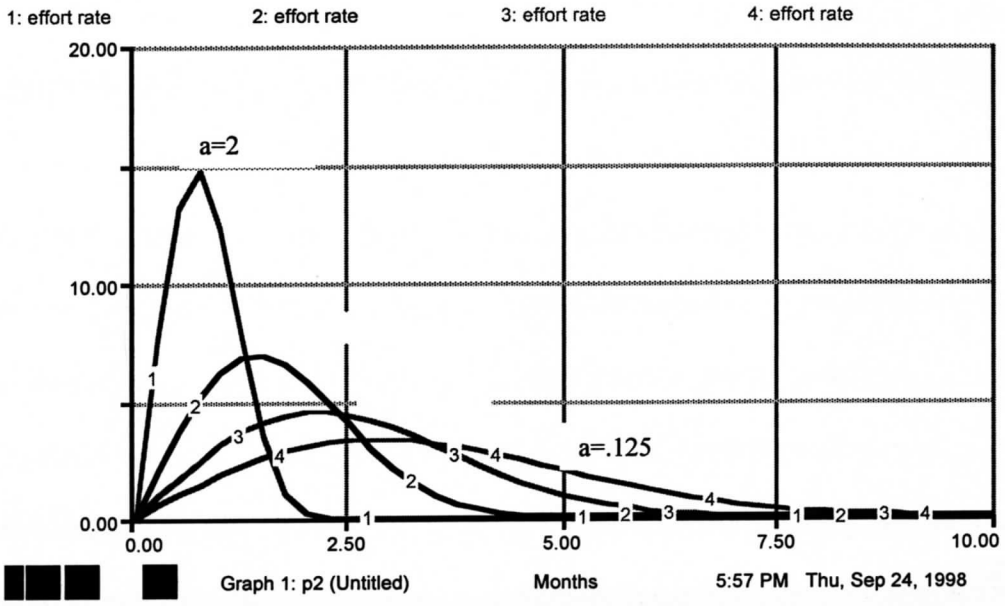

Figure 5.1a Test Results

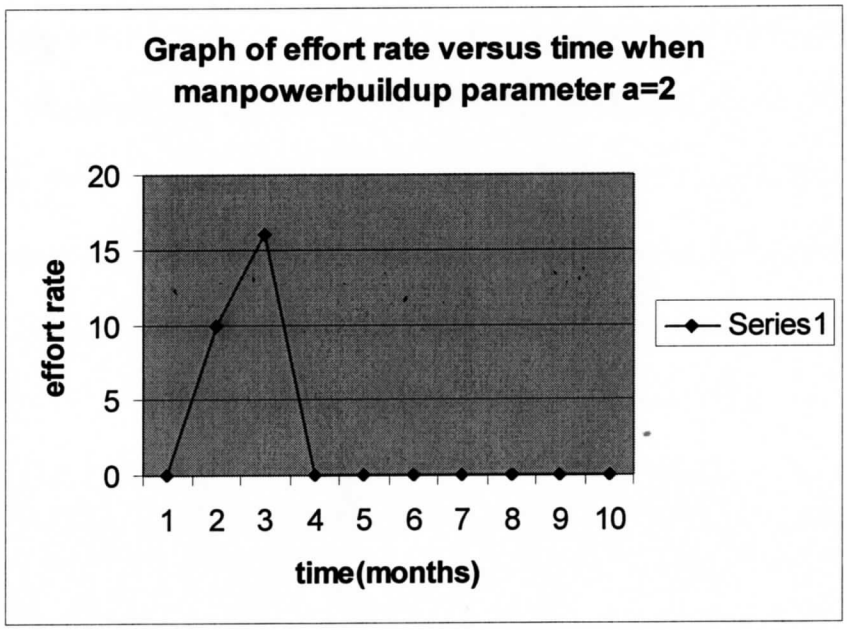

Figure 5.1b Actual results

5.3 Step 2 Evaluating and validating SoProMaTT

Step 2 aims at evaluating and validating SoProMaTT.

Submitting SoProMaTT to evaluation in a controlled experiment with students will test this. In the planned experiment, the project management related knowledge gained by the students that were trained with SoProMaTT is compared to the knowledge gained by a control group of students that were trained in a different way within the same period of time.

The Evaluation Mechanism:

For evaluating the effectiveness of SoProMaTT two groups of students will be selected. The selection is random.

- Group A: Students who will be trained with the tool.
- Group B: Students who will be trained with traditional techniques only.

Both the groups are first put to a preliminary test at the start of the semester. The test will help in determining the following information from the students.

- Their interest and awareness level of the software industry and its related problems.
- Their interest and awareness level of software project management and its related problems.

Then students of Group A will be trained in the following way:

- Project Management Introduction: This involves the basics about the use of the models in project management.

- Project Manager Role-play: This involves extensive use of SoProMaTT and performing all the activities by running the simulation models and doing the assignments.
- Feedback and learning: During the class hours the students can perform activities and ask questions to the faculty regarding the various parameters and the managerial decisions involved in a software project. The faculty can give his insights and guide the student effectively. The students also need to do the assignments and by viewing their grades the students know where they stand in the class.

Students of Group B will be trained in the following way:

- Project Management Introduction: This involves the basics about the use of different techniques in project management.
- Classroom teaching using conventional project management techniques.

At the end of the semester both the groups will be evaluated by a questionnaire, which has questions like:

- Questions about personal interest in learning more about software project management
- Questions about personal characteristics (age, gender), level of university education, practical software development experience, software project management literature background, and preferred learning style.
- Questions about typical performance patterns of software projects.
- Questions about project planning problems

- Questions on actual time consumption in learning the objectives of the various lessons.

After this both the groups will be subjected to a test. The test will determine how much each student has learnt and understood. Based on the data collected from the students after answering the questionnaire and the test results the following hypotheses will be tested.

- The average performance of both the groups after the final test is better than during the preliminary test.
- The average performance of Group A is better than that of Group B.

The first hypothesis is motivated by the fact that either group A or B will perform better after a training session. The second hypothesis is motivated by the fact that Project Management Role play will first give the students a greater feel of software projects. Their level of interest is increased. They also can respond to situations better than students of Group B without resorting to fire fighting policies.

Table 5.1 A sample questionnaire.

No.	*Question*	*Response*	*Comments*
1	Does software project management interest you and are you aware of the issues and problems facing the software industry?	Yes/No	
2	Do you have any prior experience in software development?	Yes/No	
3	What is the level of software engineering and IS education you have had till now?		
4	Do you prefer more lab work or more class lectures?	Yes/No	
5	If you have had software development experience before what development methodology did you choose to develop the system.		
6	What are the typical problems you have faced when planning any project?		
7	Does SoProMaTT help you in understanding some concepts and issues of software project management?	Yes/No	
8	Were all the objectives fulfilled by using the tool?	Yes/No	
9	How much time did it take approximately to go through each lesson?		
10	Does the tool give you enough feedback regarding the project management decisions you made while using the tool?	Yes/No	
11	Is SoProMaTT a useful teaching aide for software project management education?	Yes/No	
12	What areas do you think need improvement in terms of the tool usage and functionality?		

CHAPTER 6 - Conclusions, issues and future research

SoProMaTT has been designed and developed successfully as a teaching tool for software project management. After the first phase of testing SoProMaTT we find that it meets the criteria of being reliable and accurate by comparing the results and graphs with other simulation models. Also it is modular i.e. modules or lessons can be easily plugged into the tool.

The improvement in SoProMaTT over other approaches or other teaching tools developed comes from the following observations:

- It is completely web-enabled. The tool can be deployed on the intranet and the students can access it from anywhere within the campus.
- The tool uses several smaller simulation models rather than one single simulation model. This approach is useful as more simulation models can be incorporated into the tool in the future with each simulation model teaching the students a particular area of software project management.

SoProMaTT also meets the criterion of being user friendly. The navigation is fairly simple. Features such as the study guide, lesson notes for each lesson, and sample results make it easy for the student to use.

Although SoProMaTT meets our initially stated objectives more work needs to be done in the following areas:

- The second phase of testing i.e. the evaluation needs to be completed. The results obtained after the students actually use the tool will point out the areas of future improvements and research.

- More functionality in the form of modules needs to be added. Currently the tool supports four lessons out of which three have simulation models incorporated in them. Simulation models for risk management, cost estimation and other areas of software project management can be added.
- The tool currently does not have a database layer. A database backend needs to be built in the future. This database has to address the two issues.
    - It as to store student information in terms of the students name, ID etc.
    - It has to store the results of the activities and the assignments submitted by the students. These results can help in much better feedback by the faculty.
- Scalability is another issue that needs to be addressed in the future. Currently the results of the graphs are stored as Excel sheets. Secondly each student instantiates the extend object whenever he runs the model. This places a huge load on the server making it slow. Methods such as packaging and deploying the tool as smaller .Cab files or using multiple servers need to be looked into.

References

Tarek, Abdel-Hamid, and Stuart, E. Madnick. Software Project Dynamics An Integrated Approach. Englewood Cliffs, New Jersey: Prentice - Hall, 1991.

Brooks, Frederick P. The Mythical Man-Month. Reading, Mass: Addison-Wesley, 1995.

Dietmar, Pfahl ., Marco, Klemm, and Gunther, Ruhe. Using System Dynamics Simulation models for Software Project Management Education and Training. Fraunhofer Institute for Experimental Software Engineering (IESE), 2000.

Charette, R.N. "Large-scale Project Management Is Risk Management." IEEE Software 13.4 (1996): 110-117.

Forrester, Jay W. System Dynamics and the Lessons of 35 Years, Technical Report D-4224-4,Sloan School of Management, Massachusetts Institute of Technology, 1991.

Sycamore, Douglas M. Improving Software Project Management through System Dynamics Modeling, Master of Science Thesis, Arizona State University, 1996.

Merrill, Derek and James, Collofello. Improving Software Project Management Skills Using a Software Project Simulator, the Frontiers in Education Conference (FIE), 1997.

Appendix A – Project Details

A.1 Project Plan

In order that the project succeed its objectives a sound lifecycle process and an accompanying project plan is needed. Table A.1 shows the work breakdown structure of the project.

A.2 Project Charter

Project Objective Statement

- To develop a web-based training tool for software project management.

Roles and Responsibilities

- Sponsors: MSIS department in the BIS school of DSU
- Project Managers: Dr. Omar EL-Gayar, Dr. Terry Dennis
- Lead Analyst: Dr. Omar El-Gayar
- Designer/Architect/Programmer: Sreedhar
- Users: Faculty, Students

Business Purpose

- Teaching tool to students and aspiring project managers

Business Objectives

- Strengthen the Project Management and System Analysis and Design courses curriculum
- Make the courses more effective and interesting
- Help in attracting more students
- Potential to develop this project as a starting point for future research in the university.

Table A.1 Work Breakdown Structure

	Inception – Project Scope
1	*Stage 1*
2	Prepare project charter
3	Literature Review
4	Selection of Tool for system dynamics modeling
Milestones Achieved	Project Objective Statement defined
	Extend Suite selected for simulation
	Elaboration
5	Stage 2
6	- Selection of simulation models
	Construction
7	- Building of simulation models using Extend
8	- *Establishing a Visual Basic interface to the models*
9	- Building the data persistence layer
10	- Step 3 involves web enabling the tool.
11	- Test the tool
	Transition
12	- Using the tool for software project management course at Dakota State University.

- Possible to provide consulting and training services to industry in the future

Desired Features

- Strong course content incorporated in the tool
- Good user interface
- Ease of use and understanding
- Effective simulation model which captures the nuances of project management

Critical Success Factors

- Simple Powerful simulation model
- License required
- Commitment of Faculty
- Collaboration with industry

Constraints

- Licensing cost
- Time

Risks

- Ability to interface Extend with VB/ASP
- Ability of the model to capture the basics of project management

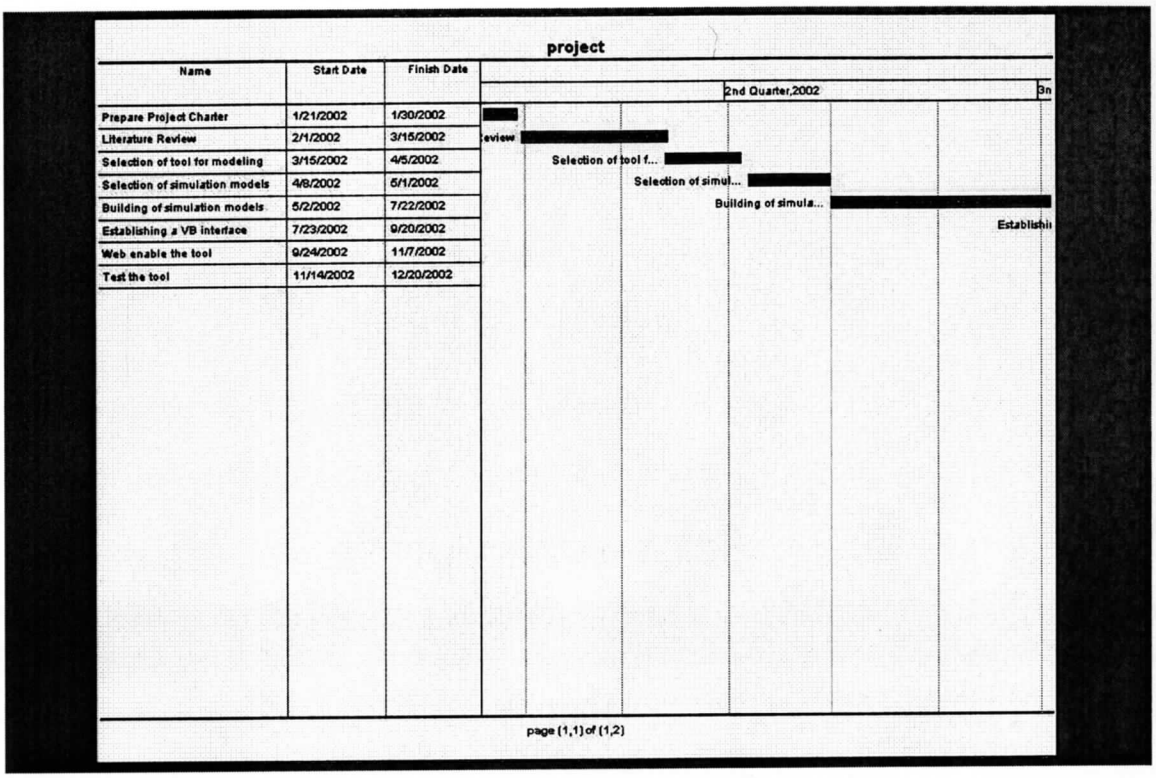

Figure A.1a Gantt chart

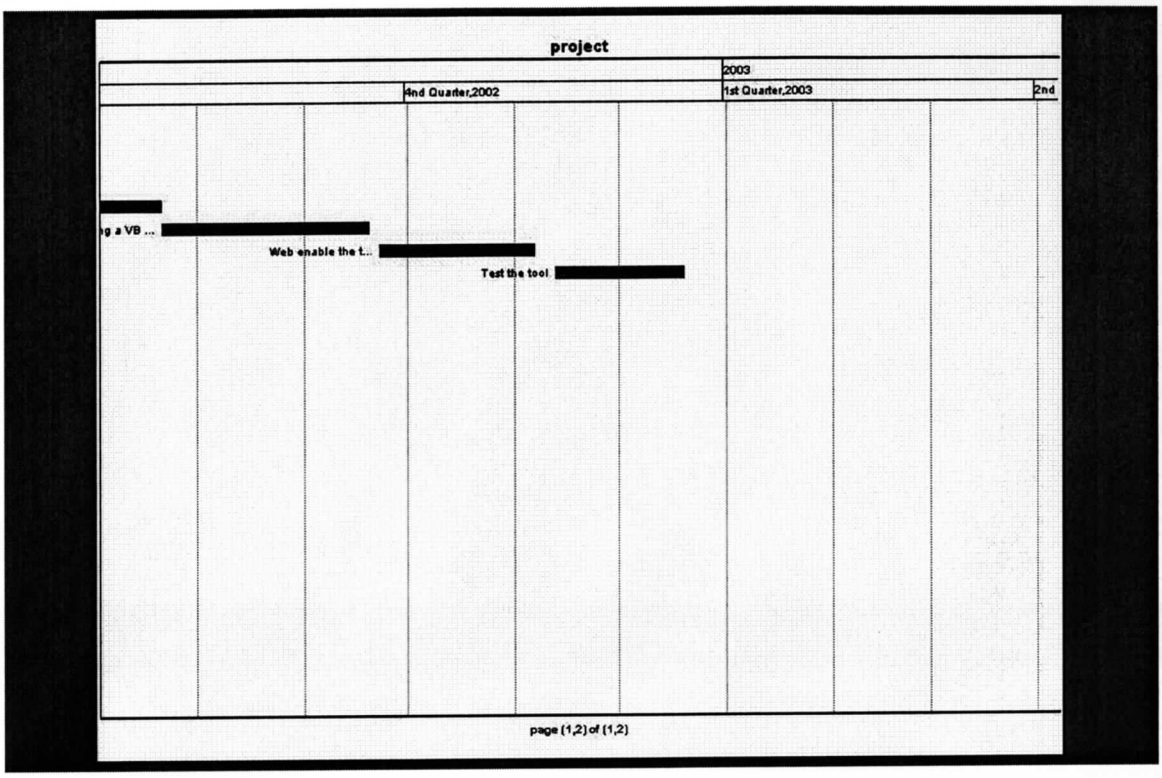

Figure A.1b Gantt chart continued

Task - Name	Start Date	Finish Date	Percent complete
Prepare Project Charter	1/21/2002	1/30/2002	100%
Literature Review	2/1/2002	3/15/2002	100%
Selection of tool for modeling	3/15/2002	4/5/2002	100%
Selection of simulation models	4/8/2002	5/1/2002	100%
Building of simulation models	5/2/2002	7/22/2002	100%
Establishing a VB interface	7/23/2002	9/20/2002	100%
Web enable the tool	9/24/2002	11/7/2002	100%
Test the tool	11/14/2002	12/20/2002	100%
Documentation and maintenance	1/1/2003	2/18/2003	100%

Figure A.1c Task details

Appendix B - Technical Guide

The tool consists of two components.

- Simulation component consisting of models built using the Extend package.
- The programming logic and user interface component built as an IIS web application with Visual Basic and ASP.

B.1 Simulation component

The simulation component consists of basically the Extend models.

There are three models presently incorporated in this tool.

- Brooks law and hiring issues model.
- Rayleigh's staffing curve and manpower model
- Quality and inspections model

Each of the three models is built using the customizable libraries present in the Extend Package. The main blocks that are used in the models are

- Holding tank
- Equation
- Constants
- Graphs
- Data send

Using these blocks the causal loops are implemented.

Data can be fed into each of the blocks through the dialog windows. This data can be fed either manually or programmatically at runtime.

B.1.1 Brooks Law Model

Each block with its corresponding equation is listed:

1. np

Nominal Productivity: The nominal_productivity: the number of Function Points a single team member can implement in a day of work (PF/person-day)This is a constant. For this model it is set to 0.1.

2. NewProjPerson

Number of new project personnel. This block is defined as a pulse function. The pulse function has three components.

- A delay: Delay = 100
- The step : a = 10
- Duration of the step: wi = 1000

The values shown above are from a sample input. It means there are 0 new personnel at the start of the project. 10 new personnel are added after 100 days into the project. The duration is the number of days these people will be on the project.

3. arate

Assimilation rate. This block is implemented as an equation.

arate = newProjPerson/30;

It is the rate in which the new personnel get assimilated into the project. In other words it takes each fresh personnel 30 days to become experienced in the project.

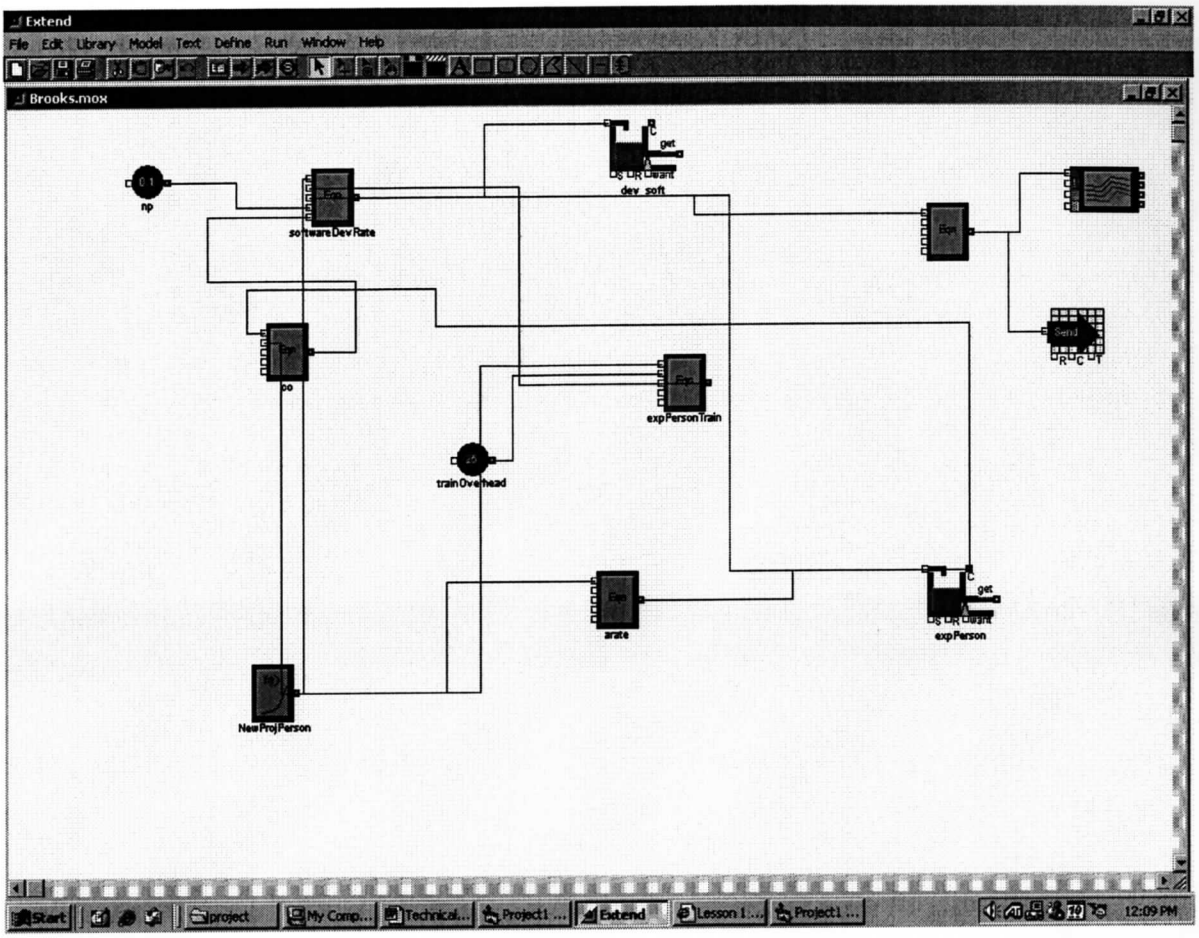

Figure B.1 Brooks Law Extend model

4. expPerson

Number of Experienced Personnel. This block is a Holding tank which stores the initial number of experienced personnel and then the number after running the model for the duration of the project.

5. trainOverhead

Training Overhead. This is implemented as a constant. The value is 25.

6. expPersonTrain

Experienced Personnel needed to train. This is an equation block.

expPersonTrain = newProjPerson *trainOverhead/100;

It means that a quarter (0.25) of an experienced person is needed to train a new person until they are fully assimilated.

7. co

Communication Overhead. This is an equation.

co=0.06*(expPerson + newProjPerson)*(expPerson + newProjPerson);

8. softwareDevRate

Software development rate. Equation

softwareDevRate =np*(1-co/100)*(0.8*newProjPerson+1.2*(expPerson-expPersonTrain));

9. Data Send Block.

This block is needed to send the graph of software development rate to an excel spreadsheet.

10. Plotter

The plotter basically plots the graph of software development rate with time.

B.1.2 Rayleigh's Model

Each block with its corresponding equation is listed:

1.manpower_build

Manpower buildup factor. This is a constant. The student can enter values between .2 - 2.

2.time

This is a straight line graph input of time for the project.

3.estimated_eff

Estimated effort left. This is a constant inputted by the user.

4.effort_rate

Effort rate. This is implemented as an equation.

effort_rate=learning_fn*(estimated_eff-cumulative_eff);

6.cumulative_eff

Cumulative effort. Holding tank.

7.learning_function

Learning function. This is an equation.

learning_fn = manpower_build*time;

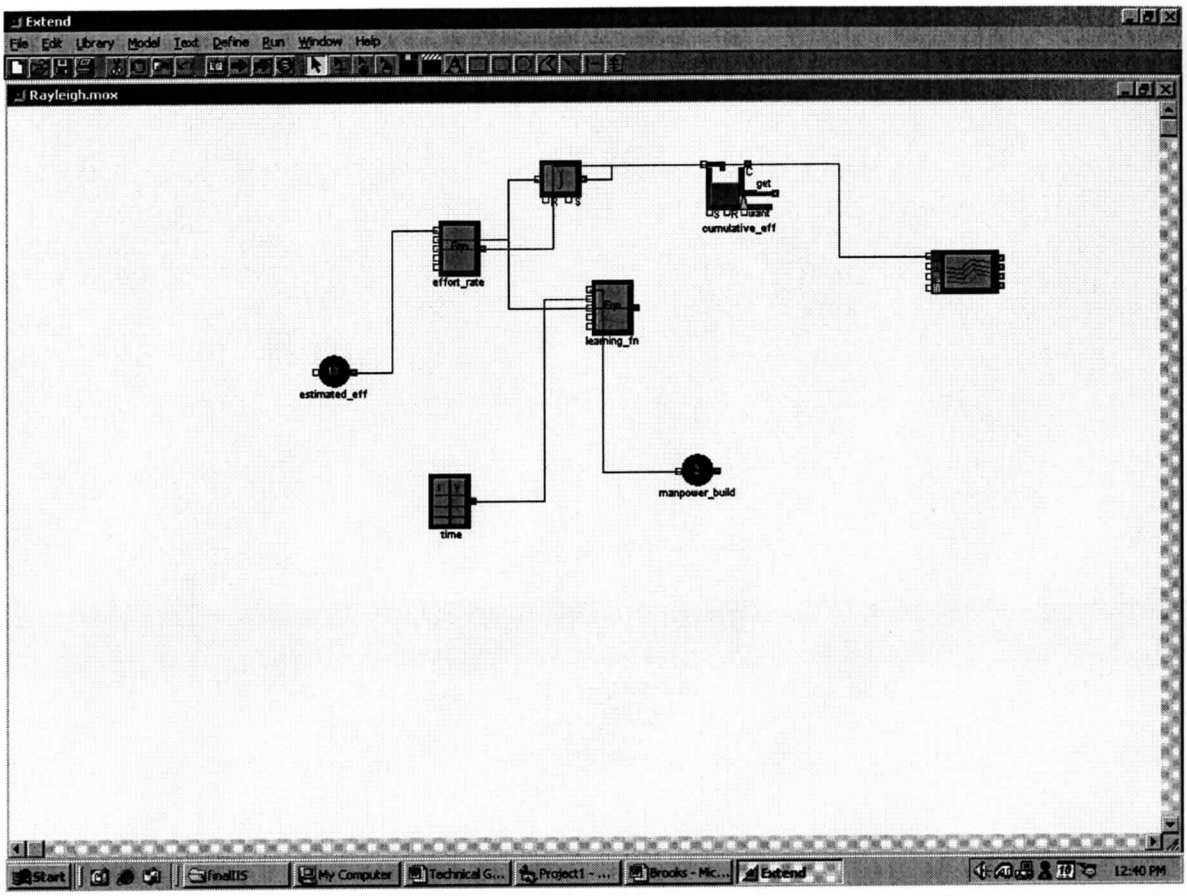

Figure B.1 Rayleigh Extend model

B.1.3 Quality Model

Equation Blocks:

CodePerDay

CodePerDay = 0.7*((Experienced*ExpRate) + (Rookie*RookieRate));

Code

Code =(CodePerDay*days);

codeinspected

codeinspected = (C/Sum)*(inspectionrate);

defects

defects = Ci/defectrate;

rework

rework = D*reworkrate;

Constant Blocks:

Experienced: Number of experienced personnel inputted by the user.

Rookie: Number of rookies inputted by the user

ExpRate: Set to 1

RookieRate: Set to 0.8

Inspectionrate: Rate of inspetctions inputted by the user. Value can be anywhere between 0 and 1. 0 indicating no inspections. 1 indicating high inspections.

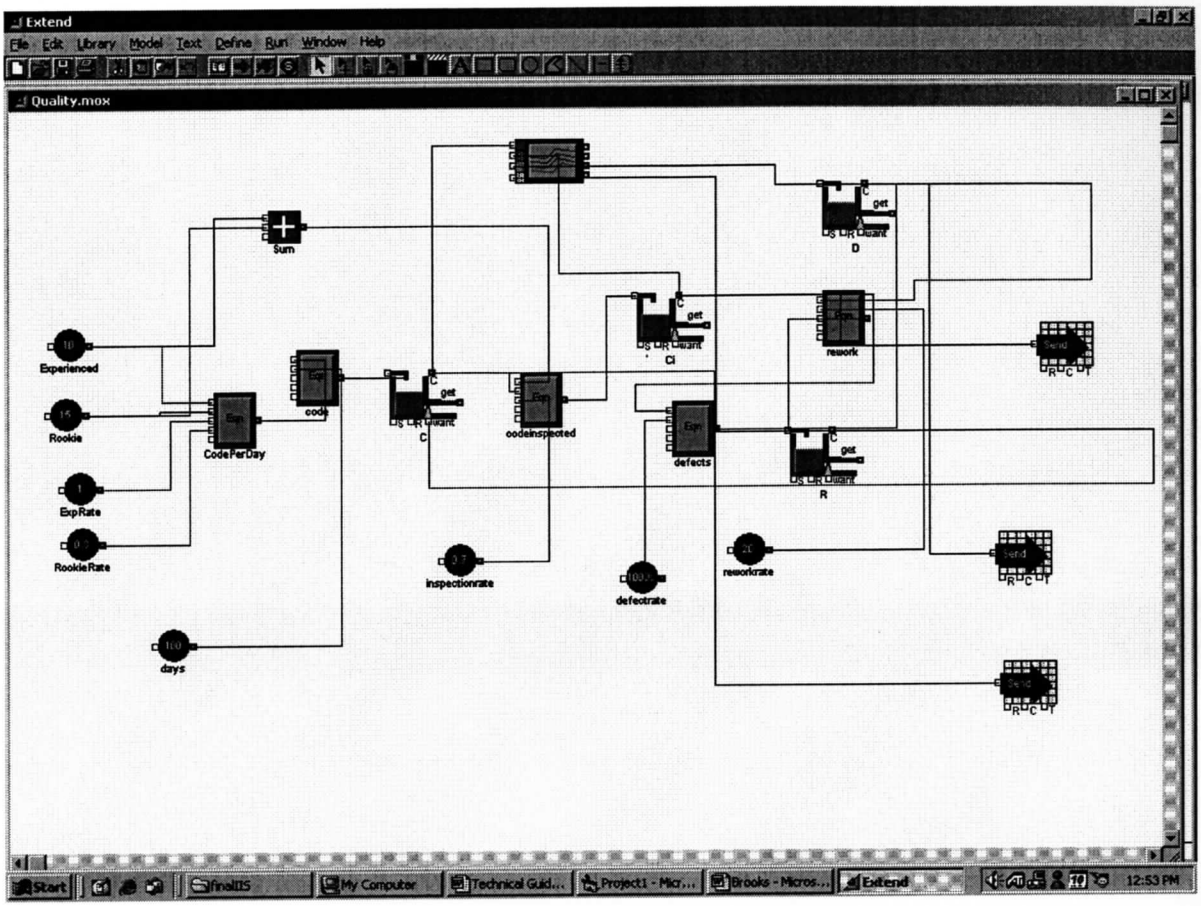

Figure B.1 Quality Extend model

B.2 Programming logic and User Interface component

The IIS application is built using VB6 and a HTML editor like Frontpage. This application consists of code modules which are compiled and run on the server. For the user interface HTML (template) pages and a Browser are used.

The main part of the IIS Application is the WebClass. The WebClass consists of one or more WebItems. WebItems contain either HTML (template) pages, Custom WebItems, or a combination of both. HTML template pages are HTML pages with special tags that are replaced by the WebClass during runtime, and HTML pages contain just plain HTML. Custom WebItems are code modules having VB functions and subroutines. The functions and procedures are called Event Procedures.

B.2.1 Building the web pages for the user interface

The following web pages will be incorporated in the tool.

- Homepage.html
- Brooks_Guide.html
- Brooks_Notes.html
- Brooks_Activities.html
- Brooks_Assignment.html
- Brooks_header.html
- Rayleigh_Guide.html
- Rayleigh _Notes.html
- Rayleigh _Activities.html
- Rayleigh_Assignment.html

- Rayleigh_header.html
- Quality_Guide.html
- Quality _Notes.html
- Quality _Activities.html
- Quality_Assignment.html
- Quality_header.html
- Footer.html

B.2.2 Building the IIS application

The steps involved are:

- Open Visual Basic 6, and choose IIS Application. A template Webclass is generated, and double clicking it makes the Webclass Designer appear on the screen.
- Save the project.
- Within the Webclass designer the HTML Template Pages are added
    - Homepage
    - Brooks_Activities
    - Rayleigh_Activities
    - Quality_Activities

The first event procedure is the WebClass_Start event. The application then starts from this page. In this tool we need the application to run from the homepage.

```
Private Sub WebClass_Start()
      Set NextItem = Homepage
 End Sub
```

The homepage of SoProMaTT has the menu from which the student can choose the lessons

he wishes to proceed to.

The next event procedure is the Respond event. Here using the writetemplate method we can display the homepage.

```
Private Sub Homepage_Respond()
    Homepage.WriteTemplate
End Sub
```

Brooks_Activities is the next HTML template. In the form event of this template the connection to the Brooks Law Extend model is made. The Extend object is instantiated through the code.

```
Private Sub Brooks_Activities_Form1()
Dim req, exp, output, days, try, rookies
' The Extend application is an object
Dim ExtendApp As Object
Dim GettingObject
Dim ExtendPath As String
'Create an instance of Extend
  On Error GoTo ErrorHandler:       ' If Extend is not loaded an error
will be returned and the CreateObject method will be called
    GettingObject = 1
   ' Get the active Extend object. If Extend is not open, this will
cause an error
    '   and will go to the ErrorHandler label. Here, the CreateObject
function is called
 Set ExtendApp = GetObject(, "Extend.Application")
loadmodel:
    GettingObject = 0
    ' Find out where Extend is installed
 ExtendApp.Execute "GlobalStr0 = GetAppPath();"        ' get the path
to extend
 ExtendPath = ExtendApp.Request("System", "GlobalStr0+:0:0:0")
' open the model
 ExtendApp.Execute "OpenExtendFile(" + """ + ExtendPath +
"C:\Documents and Settings\thotas\Desktop\MSIS\finalIIS\Brooks.mox" +
""""" + ");"
    ' Get the values from the user from the Brooks_Activities form
   req = Request.Form("req")
   exp = Request.Form("exp")
   days = Request.Form("days")
   rookies = Request.Form("rookies")

    ' Poke the value into the dialog of the block
  ExtendApp.Poke "System", "levelinit:#1:0:0", CStr(req)
 ExtendApp.Poke "System", "levelinit:#8:0:0", CStr(exp)
 ExtendApp.Poke "System", "levelinit:#2:0:0", CStr(days)
 ExtendApp.Poke "System", "levelinit:#2:1:0", CStr(rookies)
 ExtendApp.Poke "System", "levelinit:#2:2:0", "900"
```

```
 ExtendApp.Execute "RunSimulation(FALSE);"
'Request the dialog variable from Extend
 output = ExtendApp.Request("System", "contents:#1:0:0")
' Destroy the Extend object
Set ExtendApp = Nothing
       GoTo done:
ErrorHandler:

    If GettingObject Then

       ' Extend is not running (otherwise GetObject would have worked)
so we
       '   call create object to start Extend

       Set ExtendApp = CreateObject("Extend.Application")
       GoTo loadmodel:
    Else
        MsgBox Error$
        Set ExtendApp = Nothing
    End If

done:

  With Response
    .Write "<html>"
    .Write "<body>"
    .Write "<center>"
    .Write "<br>"
    .Write "<b>"
    '.Write "output = " & try
    .Write "<br>"
    .Write "<p>"
    '.Write "<a href =C:\Brooksample2.xls>Click here for the graph of
software developmant rate with time"
    .Write "<br><br>"
    .Write "<a href = http://localhost/Project1-
6/Brooks_Activities.htm>Back to Brooks activities"
    .Write "</center>"
    .Write "</body>"
    .Write "</html>"
  End With

End Sub
```

Rayleigh_Activities is the next HTML template. In the form event of this template the connection to the Rayleigh's Extend model is made. The Extend object is instantiated through the code.

```
Private Sub Rayleigh_Activities_Form1()
Dim estTotalEffort, manBuildRate, effortRate
' The Extend application is an object
```

```
Dim ExtendApp As Object
Dim GettingObject
Dim ExtendPath As String
'Create an instance of Extend
  On Error GoTo ErrorHandler:       ' If Extend is not loaded an error
will be returned and the CreateObject method will be called
    GettingObject = 1
        ' Get the active Extend object. If Extend is not open, this
will cause an error
    '   and will go to the ErrorHandler label. Here, the CreateObject
function is called

 Set ExtendApp = GetObject(, "Extend.Application")
loadmodel:
    GettingObject = 0
    ' Find out where Extend is installed
 ExtendApp.Execute "GlobalStr0 = GetAppPath();"        ' get the path
to extend
 ExtendPath = ExtendApp.Request("System", "GlobalStr0+:0:0:0")
' open the model
 ExtendApp.Execute "OpenExtendFile(" + """ + ExtendPath +
"C:\Documents and Settings\thotas\Desktop\MSIS\finalIIS\Brooks.mox" +
""""" + ");"
    ' Get the values from the user from the Brooks_Activities form
estTotalEffort = Request.Form("T1")
   manBuildRate = Request.Form("T2")
    ' Poke the value into the dialog of the block
ExtendApp.Poke "System", "levelinit:#3:0:0", CStr(estTotalEffort)
 ExtendApp.Poke "System", "levelinit:#4:0:0", CStr(manBuildRate)
  ExtendApp.Execute "RunSimulation(FALSE);"
'Request the dialog variable from Extend
 output = ExtendApp.Request("System", "contents:#1:0:0")
' Destroy the Extend object
Set ExtendApp = Nothing
       GoTo done:
ErrorHandler:
    If GettingObject Then
       ' Extend is not running (otherwise GetObject would have worked)
so we       '   call create object to start Extend
       Set ExtendApp = CreateObject("Extend.Application")
       GoTo loadmodel:
    Else
        MsgBox Error$
        Set ExtendApp = Nothing
    End If
done:
  With Response
    .Write "<html>"
    .Write "<body>"
    .Write "<center>"
    .Write "<br>"
    .Write "<b>"
    '.Write "output = " & effortRate
    '.Write "<a href =C:\ray.xls>Click here for the graph of effort
rate with time"
    .Write "<br><br>"
```

```
    .Write "<a href = http://localhost/Project1-
6/Rayleigh_Activities.htm>Back to Rayleigh Model"
    .Write "</center>"
    .Write "</body>"
    .Write "</html>"
  End With
    End Sub
```

Quality_Activities is the next HTML template. In the form event of this template the connection to the Quality Extend model is made. The Extend object is instantiated through the code.

```
Private Sub Rayleigh_Activities_Form1()
Dim qualityExp, qualityRookie, inspRate
' The Extend application is an object
Dim ExtendApp As Object
Dim GettingObject
Dim ExtendPath As String
'Create an instance of Extend
  On Error GoTo ErrorHandler:       ' If Extend is not loaded an error
will be returned and the CreateObject method will be called
    GettingObject = 1

    ' Get the active Extend object. If Extend is not open, this will
cause an error
    '   and will go to the ErrorHandler label. Here, the CreateObject
function is called

 Set ExtendApp = GetObject(, "Extend.Application")
loadmodel:

    GettingObject = 0

    ' Find out where Extend is installed

 ExtendApp.Execute "GlobalStr0 = GetAppPath();"        ' get the path
to extend
 ExtendPath = ExtendApp.Request("System", "GlobalStr0+:0:0:0")
' open the model
 ExtendApp.Execute "OpenExtendFile(" + """ + ExtendPath +
"C:\Documents and Settings\thotas\Desktop\MSIS\finalIIS\Brooks.mox" +
""""" + ");"
    ' Get the values from the user from the Brooks_Activities form
qualityExp = Request.Form("qualityExp")
 qualityRookie = Request.Form("qualityRookie")
    inspRate = Request.Form("inspRate")



    ' Poke the value into the dialog of the block
ExtendApp.Poke "System", "levelinit:#10:0:0", CStr(qualityExp)
```

```
 ExtendApp.Poke "System", "levelinit:#21:0:0", CStr(qualityRookie)
 ExtendApp.Poke "System", "levelinit:#31:0:0", CStr(inspRate)

 ExtendApp.Execute "RunSimulation(FALSE);"
'Request the dialog variable from Extend
 output = ExtendApp.Request("System", "contents:#1:0:0")
' Destroy the Extend object
Set ExtendApp = Nothing
       GoTo done:
ErrorHandler:

    If GettingObject Then

       ' Extend is not running (otherwise GetObject would have worked)
so we
       '   call create object to start Extend

       Set ExtendApp = CreateObject("Extend.Application")
       GoTo loadmodel:
    Else
        MsgBox Error$
        Set ExtendApp = Nothing
    End If

done:
With Response
    .Write "<html>"
    .Write "<body>"
    .Write "<center>"
    .Write "<br>"
    .Write "<b>"
    '.Write "<a href =C:\codedeveloped.xls>Click here for the graph of
codedeveloped with time"
    .Write "<br>"
    '.Write "<a href =C:\inspections.xls>Click here for the graph of
inspections with time"
    .Write "<br>"
    '.Write "<a href =C:\rework.xls>Click here for the graph of rework
with time"
    .Write "<br><br>"
    .Write "<a href = http://localhost/Project1-
6/Quality_Activities.htm>Back to Quality Model"
    .Write "</center>"
    .Write "</body>"
    .Write "</html>"
  End With
    End Sub
```

The next step is to compile the project.

Compiling the project generates an ASP start page, and a Webclass run-time dll.

The last step is deploying the application

You deploy the application using the Package and Deployment Wizard.

The steps involved in building SoProMaTT are:

- First build the simulation models using the Extend package
- Build the web pages using any HTML editor
- Implement the programming logic and interface the web pages to the Extend models using the IIS web application template of Visual Basic 6.0.
- Compile the application as a dll.
- Package the application as a web package. Add the HTML pages and the Extend models to the web package.
- Deploy the application on the web server

B.2.3 Adding a model

To add any models to SoProMaTT the following steps have to be taken:

- Build any new models using Extend. Save the model in the same folder as the Vb/IIS application.
- Build the HTML page having the form which accesses the Extend model and runs it.
- Add this HTML page as a template to the existing tool.
- In the form submit event of this web page implement code with the following algorithm to interface with the Extend package.

Get the path to extend

Open the model

Get the values from the user from the web form

Poke the value into the dialog of the block

Request the dialog variable from Extend

Destroy the Extend object

Using the response method display the results.

Sample code

```
Private Sub Brooks_Activities_Form1()
Dim req, exp, output, days, try, rookies
' The Extend application is an object
Dim ExtendApp As Object
Dim GettingObject
Dim ExtendPath As String
'Create an instance of Extend
  On Error GoTo ErrorHandler:       ' If Extend is not loaded an error
will be returned and the CreateObject method will be called
    GettingObject = 1

    ' Get the active Extend object. If Extend is not open, this will
cause an error
    '   and will go to the ErrorHandler label. Here, the CreateObject
function is called

 Set ExtendApp = GetObject(, "Extend.Application")
loadmodel:

    GettingObject = 0

    ' Find out where Extend is installed

 ExtendApp.Execute "GlobalStr0 = GetAppPath();"        ' get the path to
extend
 ExtendPath = ExtendApp.Request("System", "GlobalStr0+:0:0:0")
' open the model
 ExtendApp.Execute "OpenExtendFile(" + """ + ExtendPath + "C:\Documents
and Settings\thotas\Desktop\MSIS\finalIIS\Brooks.mox" + """ + ");"
    ' Get the values from the user from the Brooks_Activities form
   req = Request.Form("req")
   exp = Request.Form("exp")
   days = Request.Form("days")
   rookies = Request.Form("rookies")
    ' Poke the value into the dialog of the block
  ExtendApp.Poke "System", "levelinit:#1:0:0", CStr(req)
 ExtendApp.Poke "System", "levelinit:#8:0:0", CStr(exp)
 ExtendApp.Poke "System", "levelinit:#2:0:0", CStr(days)
 ExtendApp.Poke "System", "levelinit:#2:1:0", CStr(rookies)
 ExtendApp.Poke "System", "levelinit:#2:2:0", "900"
 ExtendApp.Execute "RunSimulation(FALSE);"
'Request the dialog variable from Extend
```

```
 output = ExtendApp.Request("System", "contents:#1:0:0")
' Destroy the Extend object
Set ExtendApp = Nothing
       GoTo done:
ErrorHandler:
    If GettingObject Then
       ' Extend is not running (otherwise GetObject would have worked)
so we       '   call create object to start Extend
    Set ExtendApp = CreateObject("Extend.Application")
       GoTo loadmodel:
    Else
        MsgBox Error$
        Set ExtendApp = Nothing
    End If
done:
  With Response
    .Write "<html>"
    .Write "<body>"
    .Write "<center>"
    .Write "<br>"
    .Write "<b>"
    '.Write "output = " & try
    .Write "<br>"
    .Write "<p>"
    '.Write "<a href =C:\Brooksample2.xls>Click here for the graph of
software developmant rate with time"
    .Write "<br><br>"
    .Write "<a href = http://localhost/Project1-
6/Brooks_Activities.htm>Back to Brooks activities"
    .Write "</center>"
    .Write "</body>"
    .Write "</html>"
  End With
End Sub
```

B.2.4 Modifying or adding a HTML page

To modify any web pages or the navigation of the tool follow these steps:

If any new pages have to be added to the tool then design the web pages and add the links to them from the other existing pages of SoProMaTT.

If the new page needs to be interfaced to and Extend model then add the page as a HTML template to the IIS web application. If the web page is a passive page i.e. a page that does not interact with the Extend model then it need not be added as a template. However the links have to be made.

B.2.5 Packaging and deploying SoProMaTT

After you have developed and debugged (you can use the full VB development environment for that) it's time to deploy. Make sure the server has the Posting Acceptor installed before you begin, and then

Follow these three steps:

- Compile.
- Use the Package and Deployment Wizard to build a .CAB (cabinet) file that contains the necessary files for the application.
- Use the same wizard to deploy your application to the Web server of your choice. This will unpack the .CAB file on the server and will install and register the necessary components. You must either deploy to an existing virtual directory on the server, or manually create one before deploying the .CAB file.

Appendix C

C.1 User Manual

When the student accesses SoProMaTT, the menu is presented to him.

The menu consists of four options that are basically lessons, which the students can access. The functionality of the tool will be in the form of these lessons. Each Lesson gives an insight into certain aspects of project management.

The lessons are as follows:

Lesson 1: **Project Management Concepts**

Lesson 2: **Brook's Law and hiring issues**

Lesson 3: **Rayleigh Model**

Lesson 4: **Quality Assurance model**

Lesson 1 has web pages that contain information about project management principles, definitions and the various activities of software project management.

Lessons 2, 3, and 4 have a similar structure. They are divided into four parts:

1. Study Guide
2. Lesson notes
3. Activities
4. Assignments

The study guide consists of the path that the student needs to take for achieving his objectives for that chapter. Lesson notes explain the chapter in more detail. They also help the students understand the various options and the consequences involved.

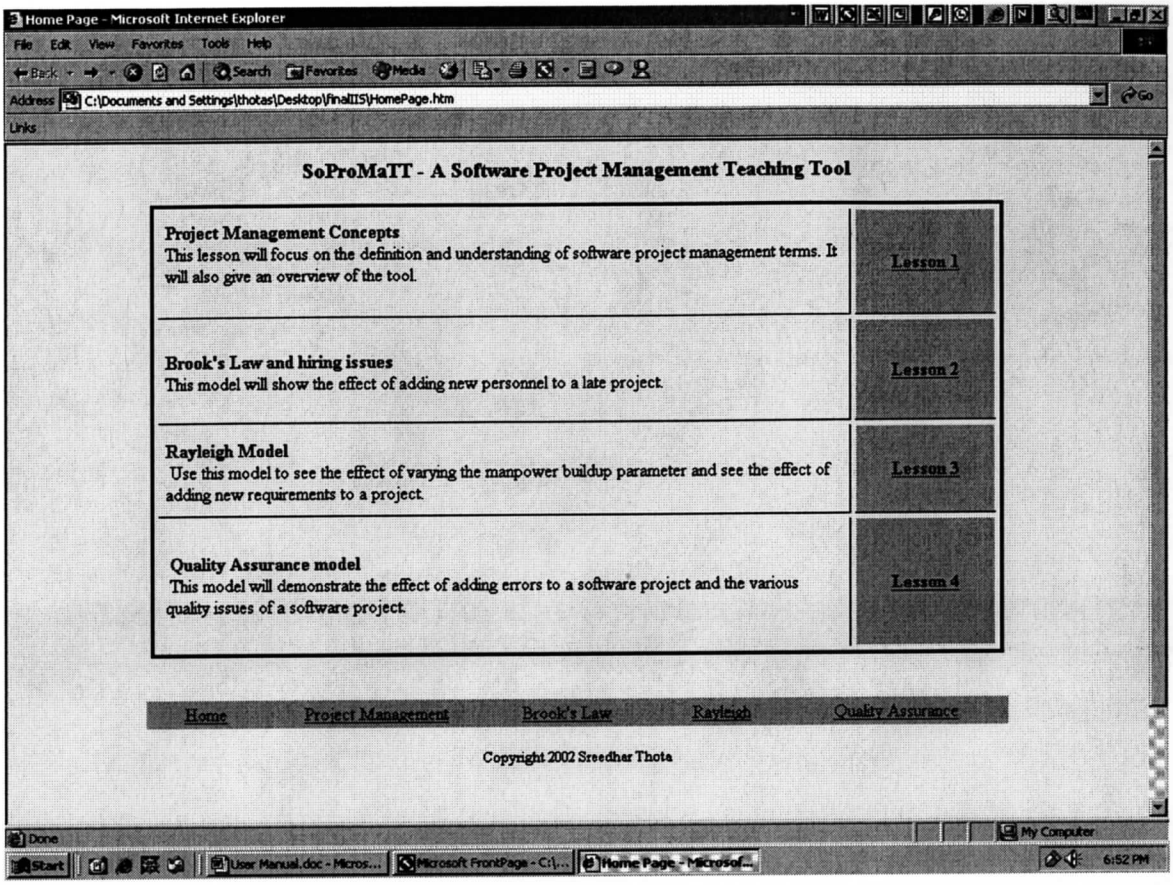

Figure C.1 Main menu screenshot

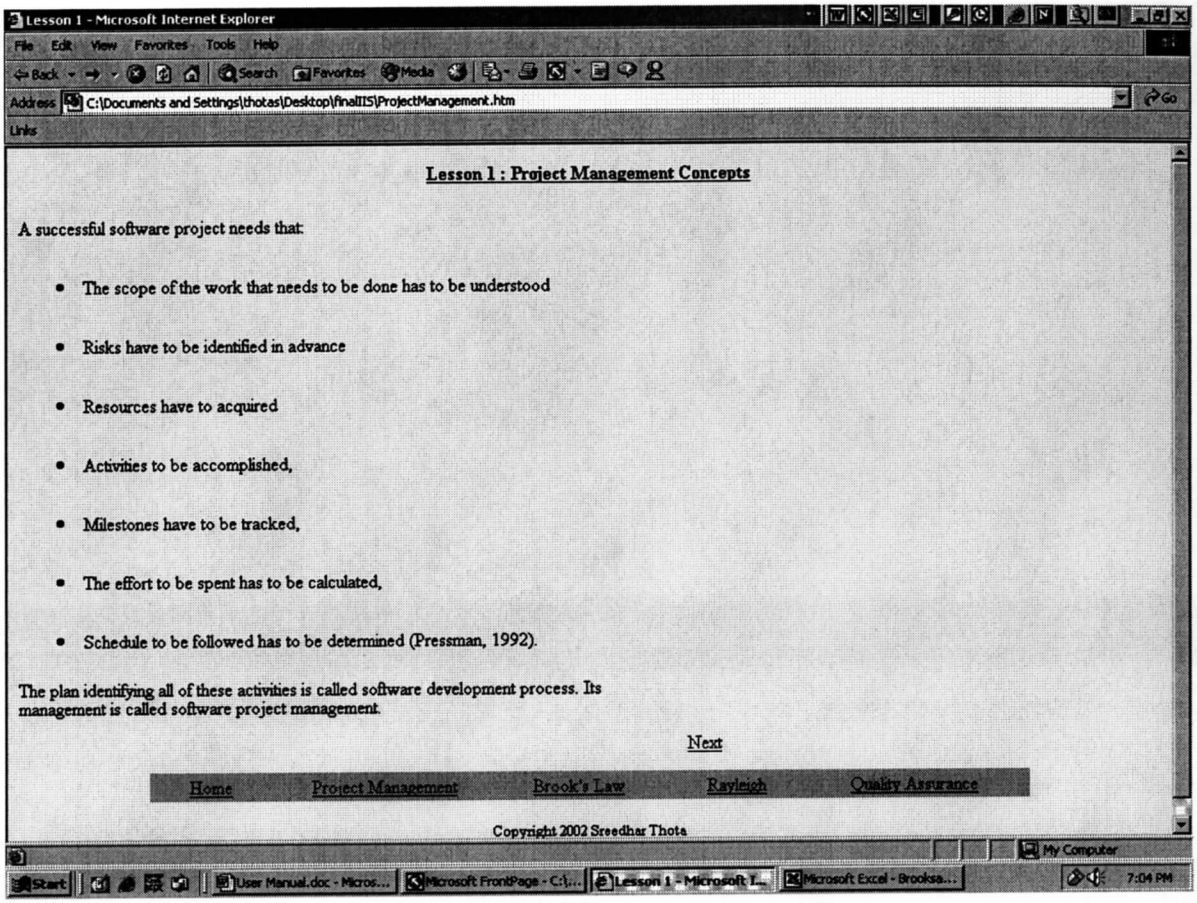

Figure C.2 Project management screenshot

The activities let the student play with the models. They assume the role of project managers and this helps them gauge the consequences of their actions. When the student inputs the required fields and submits the simulate button the extend model is invoked and the model generates the output. A sample output is shown below in figure C.5. The assignments section lists the assignments the student has to complete in order to finish the lesson.

Sample input for Brooks law model

Initial Requirements	500 function points
Experienced Personnel	20 people
After how many days are rookies added?	100 days
How many rookies?	15

Sample Input for Quality Assurance model

Experienced Personnel	10
Rookies	5
Inspection rate	0.7 (0 for no inspections and 1 for high level of inspections)

Sample Input for Quality model

Estimated total effort	15
Manpower buildup rate	0.5

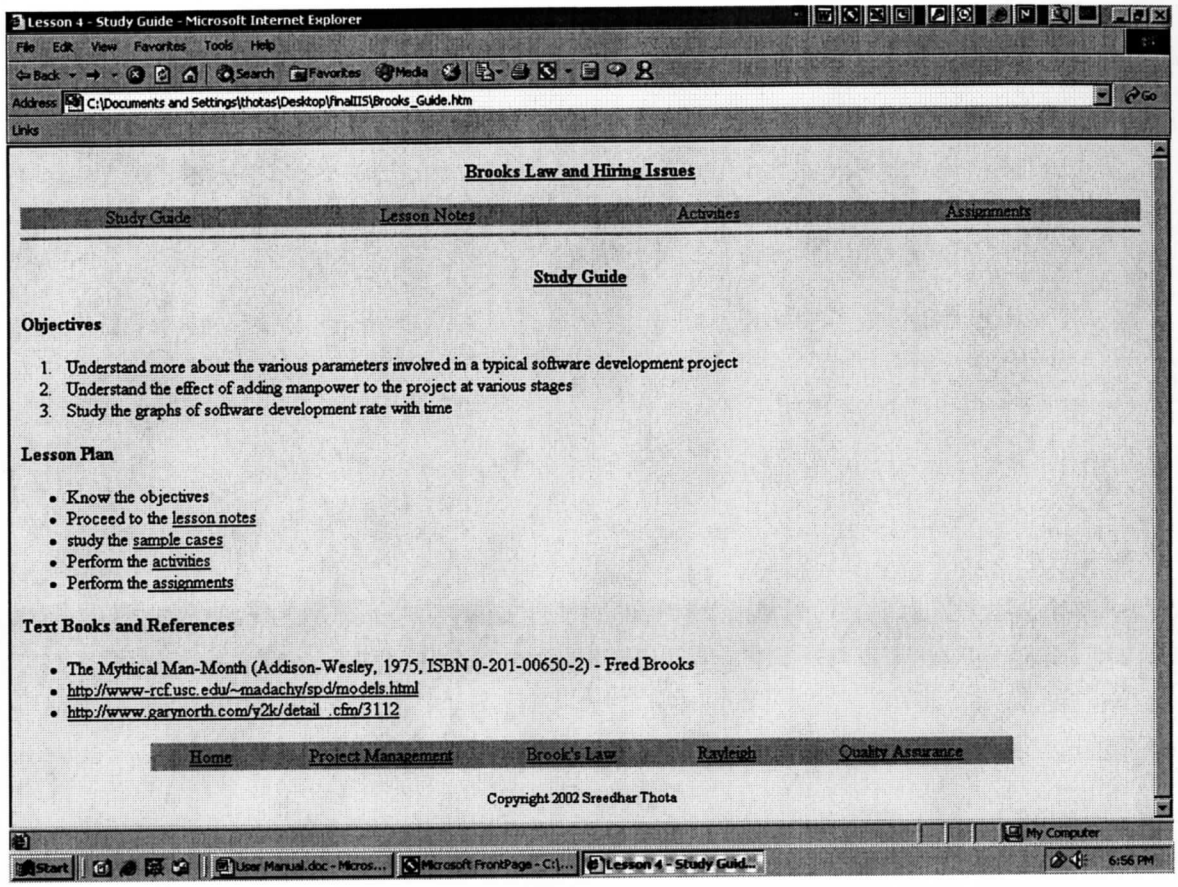

Figure C.3 Brooks law study guide screenshot

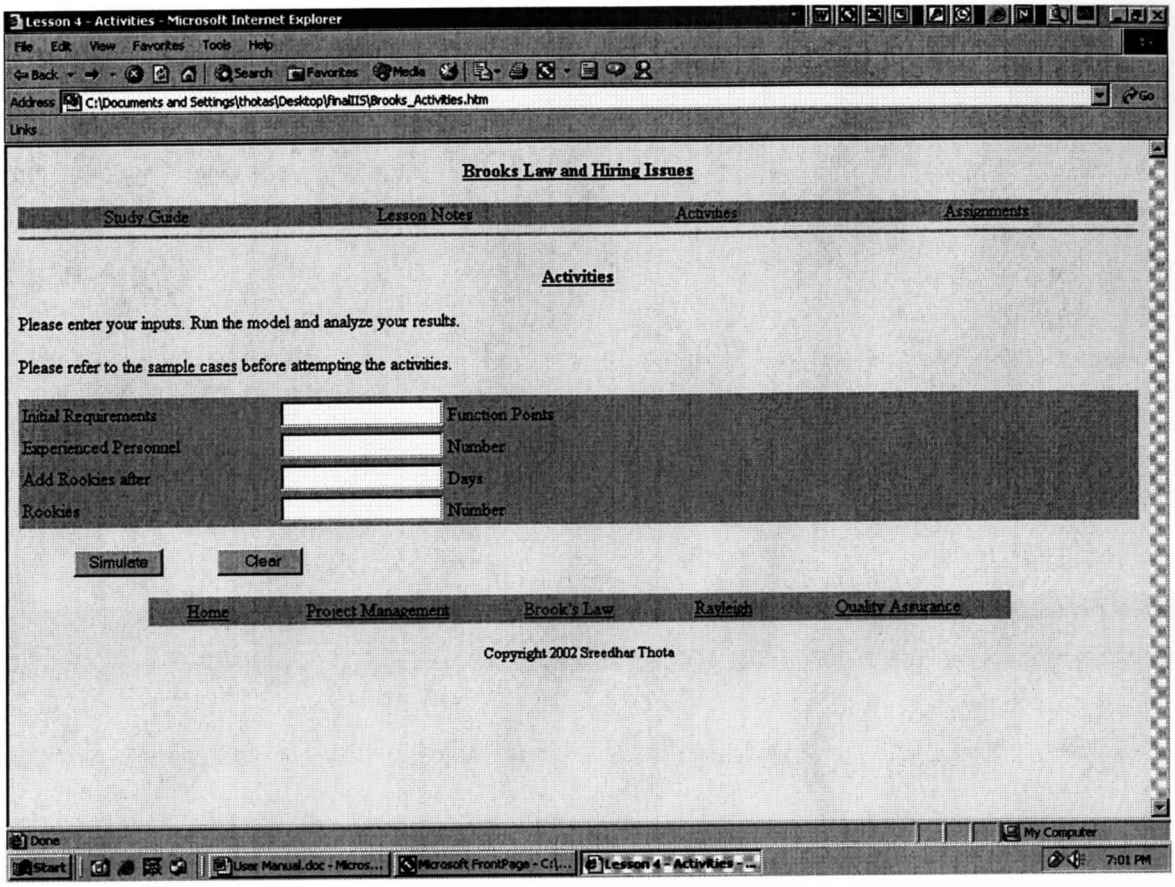

Figure C.4 Brooks law activities screenshot

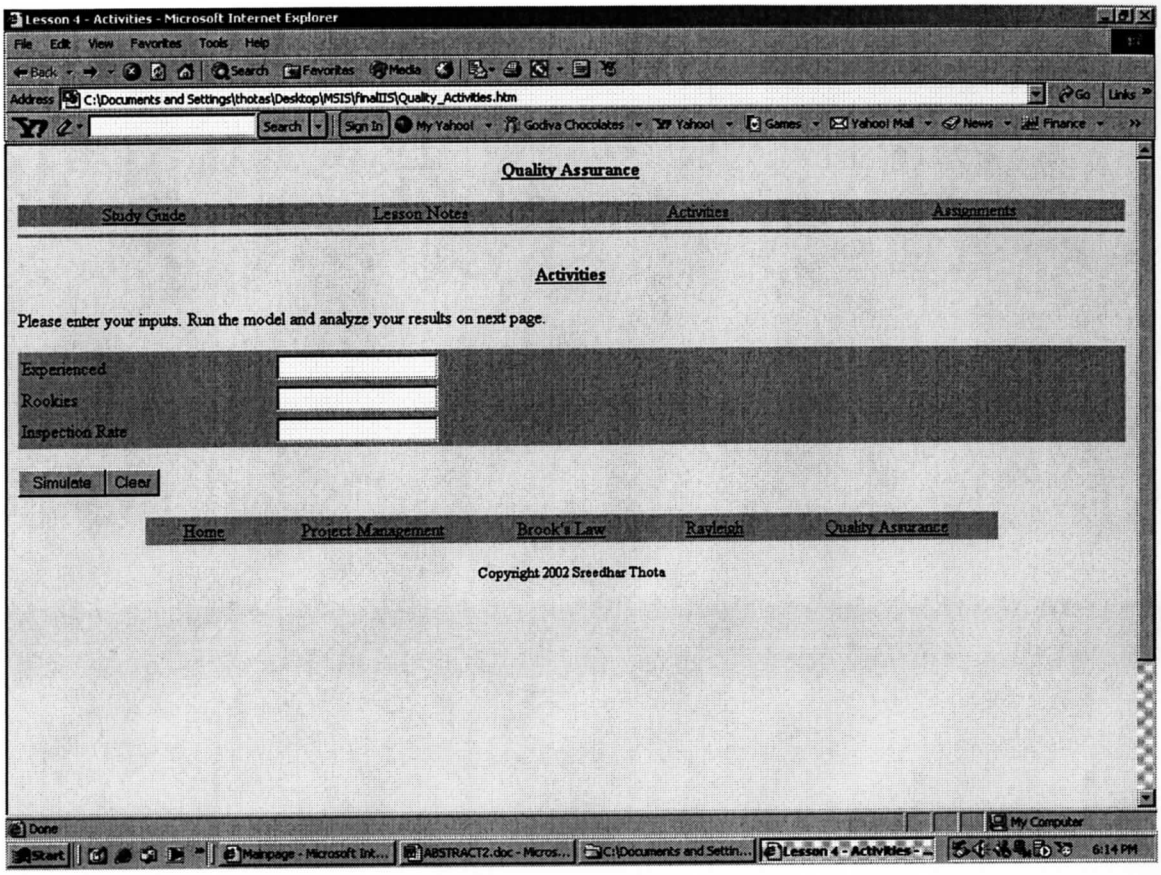

Figure C.5 Quality Assurance activities screenshot

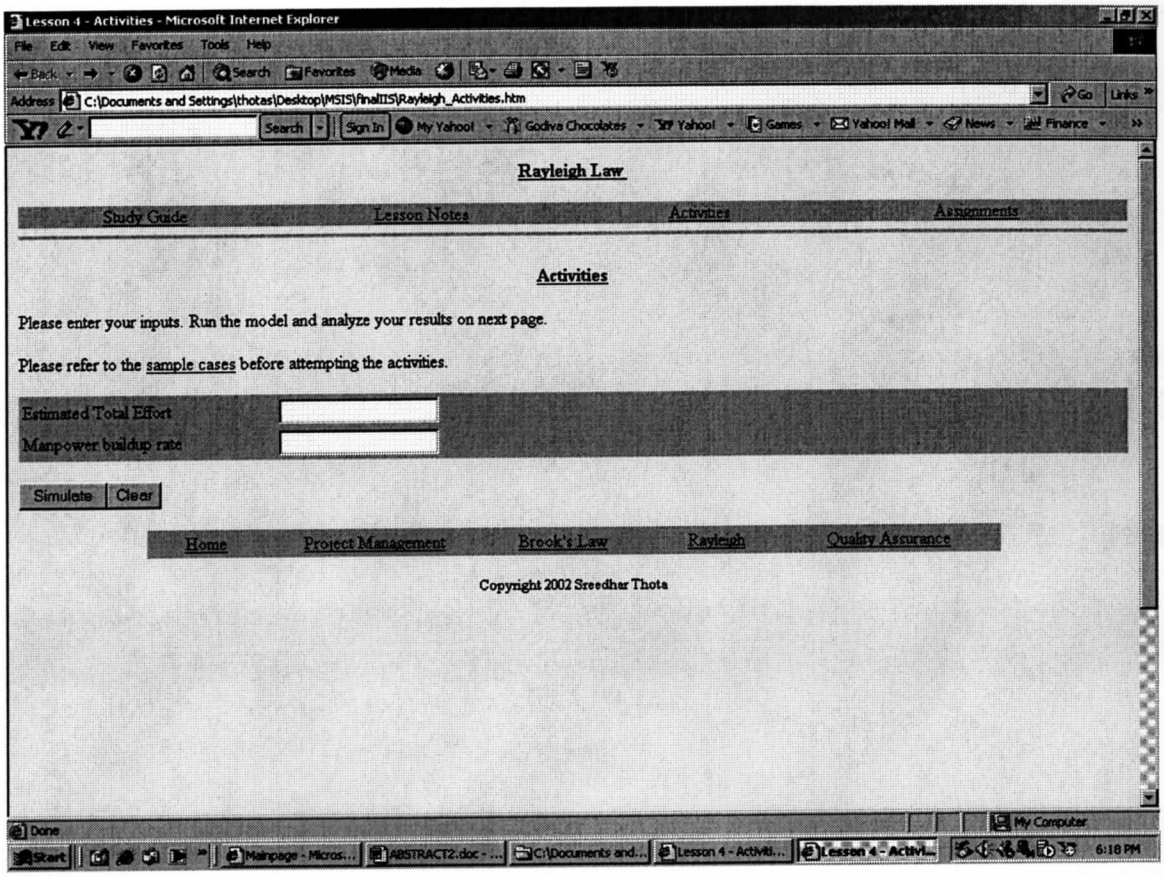

Figure C.6 Quality Assurance activities screenshot

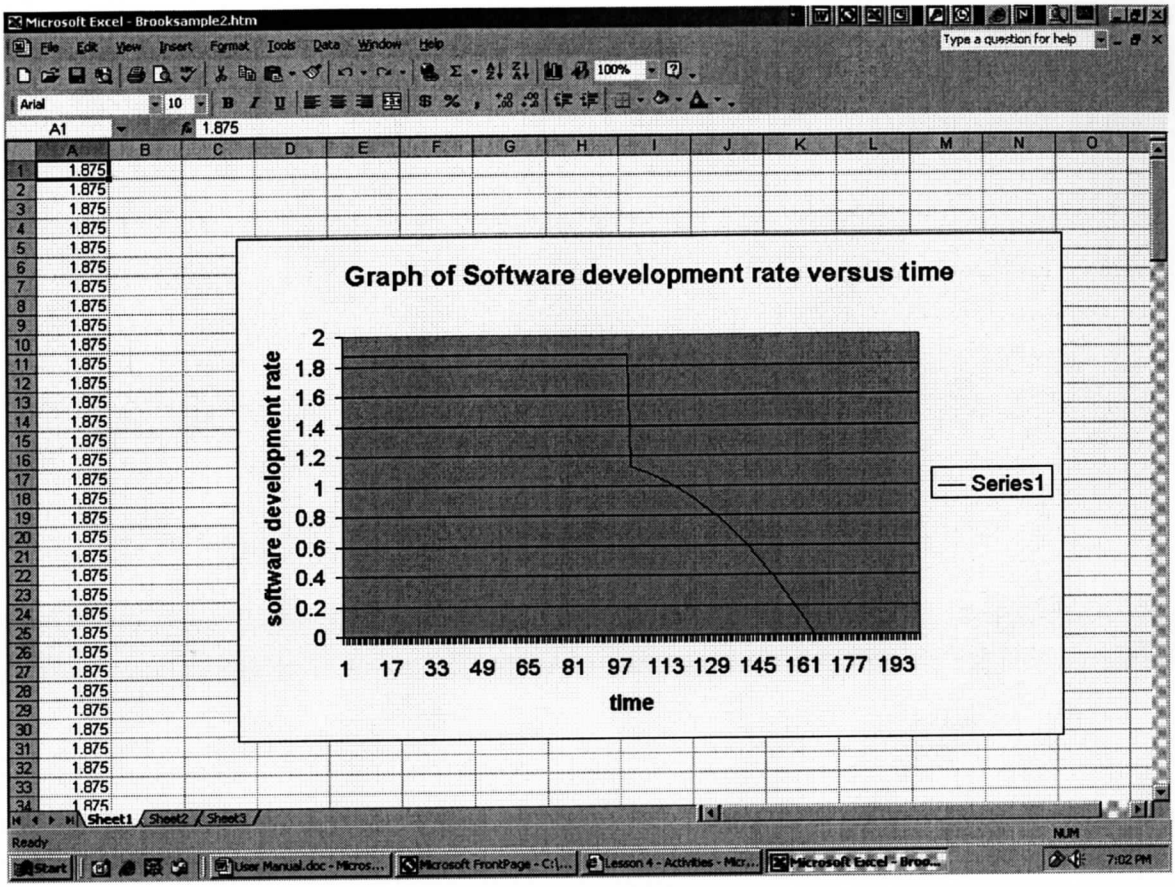

Figure C.5 Sample output screenshot